Dissertation Proposal

Exploring the Security Landscape of AR/VR Applications: A Multi-Dimensional Perspective

Abdulaziz Alghamdi

Date: November 7, 2024

Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816

Doctoral Committee:
Dr. David Mohaisen (Chair)
Dr. Murat Yuksel
Dr. Fan Yao
Dr. Hao Zheng
Dr. Yao Li

Abdulaziz Alghamdi

Department of Electrical and Computer Engineering, University of Central Florida (UCF)
4000 Central Florida Blvd., Orlando, FL 32816-2362 USA

EDUCATION

PH.D. IN COMPUTER ENGINEERING (2021 – CURRENT)
University of Central Florida

M.SC. IN ELECTRICAL AND COMPUTER ENGINEERING (2015 – 2017)
Howard University

B.SC. IN COMPUTER ENGINEERING (2004 – 2009)
Umm Al-Qura University

PUBLICATIONS

- **Abdulaziz Alghamdi**, Ali Alkinoon, Ahod Alghuried, and David Mohaisen, *xr-droid: A Benchmark Dataset for AR/VR and Security Applications*. *Accepted at* IEEE Transactions on Dependable and Secure Computing, **IEEE TDSC**, 2024.
- **Abdulaziz Alghamdi** and David Mohaisen, *Through the Looking Glass: LLM-Based Analysis of AR/VR Android Applications Privacy Policies*. *Accepted at* 23rd International Conference on Machine Learning and Applications, **ICMLA 2024**, Miami, Florida, USA, December 18-20.
- **Abdulaziz Alghamdi**, Ali Alkinoon, Ahod Alghuried, Soohyeon Choi and David Mohaisen, *Unified API Call-based Detection of Android and IoT Malware*. *Submitted to* The 40th ACM/SIGAPP Symposium On Applied Computing, **ACM SAC 2025**, Sicily, Italy, March 31 - April 4, 2025.
- Ahod Alghureid, Ali Alkinoon, **Abdulaziz Alghamdi**, and and David Mohaisen, *Simple Tricks, Big Threats: How Simple Perturbations Evade ML-based Ethereum Phishing Detection*. *Submitted to* Elsevier Computer Networks, **COMNET**, 2024.
- Ali Alkinoon, Ahod Alghuried, **Abdulaziz Alghamdi**, Soohyeon Choi and David Mohaisen, *Pandora's Box Reopened: New Insights into Android Permissions*. *Submitted to* The 40th ACM/SIGAPP Symposium On Applied Computing, **ACM SAC 2025**, Sicily, Italy, March 31 - April 4, 2025.

- Soohyeon Choi, Ahod Alghuried, Ali Alkinoon, **Abdulaziz Alghamdi** and David Mohaisen, *On Attributing ChatGPT-Transformed Synthetic Code*. *Submitted to* The 40th ACM/SIGAPP Symposium On Applied Computing, **ACM SAC 2025**, Sicily, Italy, March 31 - April 4, 2025.
- Ali Alkinoon, **Abdulaziz Alghamdi**, Ahod Alghuried and David Mohaisen, *Troid: Temporal and Cross-Sectional Android Dataset and Its Security Applications*. *Submitted to* The 40th ACM/SIGAPP Symposium On Applied Computing, Sicily, **ACM SAC 2025**, Italy, March 31 - April 4, 2025.

Contents

Abstract

The rapid evolution of Augmented Reality (AR) and Virtual Reality (VR) technologies on various consumer platforms, including mobile devices, has significantly impacted the digital landscape. This surge in AR/VR applications has not only expanded the boundaries of virtual experiences but also intensified scrutiny of their security and privacy aspects. As these technologies become deeply integrated into everyday activities, understanding their underlying security infrastructure and privacy policies becomes crucial for safeguarding user data against evolving cyber threats.

In response to the burgeoning interest in AR/VR technologies and the accompanying security concerns, our first study delves into the analysis of these applications from a security and performance standpoint. Despite the increasing popularity of AR/VR applications, there is a notable absence of benchmark datasets that facilitate comprehensive security research. To bridge this gap, we have compiled a dataset comprising 408 diverse AR/VR applications from the Google Play Store. This dataset is enriched with various data modalities such as control flow graphs, strings, functions, permissions, API calls, hexdump, and metadata. This dataset is poised to support a multitude of research endeavors, providing a foundational tool for enhancing application security across platforms.

Building on the need for enhanced privacy measures within AR/VR environments, the second study utilizes BERT, a leading-edge text classification model, to scrutinize AR/VR applications' privacy policies. Our comparative analysis of free and premium content websites reveals that AR/VR applications typically exhibit a higher percentage of clear and thorough privacy segments than free content websites. However, they fall short of premium websites' standards. The strategic emphasis on critical privacy practices and key terms within these policies aims to bolster their effectiveness. This offers insights into AR/VR privacy dynamics.

Finally, addressing the critical challenge of malware detection on digital platforms, our third study introduces an advanced, scalable approach using machine learning models, specifically Random Forest (RF) and Graph Neural Networks (GNN). This research utilizes two datasets, one containing Android applications—including AR/VR applications—and the other comprising Executable and Linkable Format (ELF) files, both featuring benign and malicious samples.

1 Introduction

1.1 Motivation

Augmented Reality (AR) and Virtual Reality (VR) technologies have swiftly evolved from niche innovations into essential tools that impact various sectors, including entertainment, education, healthcare, and industry. While these technologies enhance interactivity and provide immersive experiences, they also introduce significant challenges to security and privacy.

As AR/VR technologies become more accessible and integrate deeper into everyday applications, security vulnerabilities and privacy risks escalate. These platforms, due to their immersive nature and extensive data collection capabilities, are particularly susceptible to malicious activities. The research focuses on the foundational aspects of AR/VR security, aiming to identify and mitigate vulnerabilities inherent to these rapidly advancing technologies.

A critical area of vulnerability is the application frameworks and development platforms used in AR/VR settings, which often lack sufficient security protocols. The investigation will assess these frameworks to pinpoint common vulnerabilities and suggest strategies to secure them against security breaches.

Moreover, privacy policies associated with AR/VR technologies often lack clear communication regarding the extent and use of data collection. This leads to user distrust and regulatory issues. The analysis will compare AR/VR privacy policies against digital application norms and propose frameworks for more transparent and user-focused privacy practices.

Additionally, the prevalence and mechanics of malware in AR/VR systems will be examined to understand how these harmful programs exploit specific features of AR/VR technologies. By studying malware distribution across various platforms and infrastructures, strategies will be developed to detect and neutralize these threats effectively.

Exploring network vulnerabilities within AR/VR systems is essential for developing robust security measures. This research will analyze how infrastructural configurations within AR/VR environments influence their vulnerability to attacks. It will also devise strategies that enhance network resilience while minimizing disruptions. By addressing these key security and privacy challenges, the research aims to contribute significant advancements to AR/VR, ensuring these technologies can be utilized safely and effectively across diverse applications.

1.2 Statement of Research and Contributions

In the evolving landscape of digital technology, AR and VR technologies present both revolutionary opportunities and significant security challenges. As these technologies permeate more aspects of everyday life, they become critical fronts in the battle for security and privacy. This research aims to address multifaceted security and privacy issues emerging from AR/VR technologies adop-

tion. By exploring diverse aspects ranging from application security and privacy policy analysis to advanced malware detection, this work seeks to contribute significantly to the understanding and enhancement of AR/VR security frameworks. Our comprehensive approach spans the creation of benchmark datasets, evaluation of privacy practices, and the development of robust machine learning models to detect and classify malware effectively.

AR/VR Application Dataset and Security Analysis (§ 3). The first facet of our research focuses on the critical need for comprehensive benchmark datasets in AR/VR application security research. Given the dearth of resources for robust security and performance analysis in this rapidly expanding field, we have developed a dataset consisting of 408 AR/VR applications from Google Play Store. This dataset, standardized across multiple data modalities (*i.e.,* control flow graphs, strings, functions, permissions, API calls, hexdump, and metadata) serves as a foundational tool for various security-related research. This work fills a significant gap in current research resources. It also sets the stage for future studies aimed at enhancing AR/VR applications' security mechanisms.

Privacy Policy Evaluation in AR/VR Android Applications (§ 4). Building on the necessity for rigorous privacy standards in AR/VR applications, the second part of our research evaluates the clarity and thoroughness of privacy policies within this domain. Utilizing BERT, a cutting-edge text classification model, we analyze and compare AR/VR privacy policies to those of both free and premium content websites. Our findings reveal nuanced insights into privacy practices in the AR/VR industry. They highlight areas where AR/VR applications lag behind premium services but excel against free content offerings. This study underscores the strategic importance of clear and effective privacy policies in enhancing user trust and compliance in AR/VR environments.

Malware Detection in AR/VR and General Android Applications (§ 5). The final component of our research addresses the pressing challenge of malware detection within AR/VR and broader Android platforms. Leveraging advanced machine learning techniques, including Random Forest and Graph Neural Networks, this study utilizes two datasets—one comprising Android applications with AR/VR features and another featuring ELF binaries—to classify and analyze benign versus malicious software samples. Our approach demonstrates GNNs' superior capabilities over traditional methods. This is particularly in their ability to discern complex patterns and relationships within data, leading to markedly improved malware detection rates. This work not only enhances our understanding of malware dynamics but also contributes to the development of more secure digital environments on Android platforms.

2 Related Work

In the dynamic field of augmented and virtual reality, the exploration of security, privacy, and malware detection stands as a pivotal area of research. This body of work critically examines the advancements and challenges within these domains, underscoring the necessity for rigorous analysis and innovative methodologies. Previous studies have laid foundational work by addressing diverse aspects such as dataset creation, privacy policy evaluation, and malware detection techniques across various platforms. However, there remains a significant gap in comprehensive, targeted research specifically tailored to AR/VR applications. This section reviews the spectrum of existing literature, ranging from the development of specialized AR/VR datasets to the application of advanced machine learning models like BERT and Graph Neural Networks in privacy and security analysis. Each study contributes uniquely to the overarching goal of enhancing the robustness and understanding of AR/VR technology's security and privacy landscape.

2.1 Security and Datasets in AR/VR

AR/VR Datasets. Although limited in scope and purpose, several works exist on compiling datasets for AR/VR software analysis. For instance, Rzig *et al.* [70] and Nusrat *et al.* [64] curated applications built using the unity framework—a widely-used game engine offering a robust environment for developing both 2D and 3D games, interactive simulations, and other graphical content [81]—for software analysis. However, the main purpose of the studies is not benchmark dataset: while Rzig *et al.* [70] focused on understanding the prevalence, quality and effectiveness of testing practices in 314 open-source VR apps, Nusrat *et al.* [64] focused on understanding optimization practices, e.g., simplifying graphics, rendering, language, and APIs, using 100 VR applications.

While essential prior work, those studies fall short in their diversity of the AR/VR applications, the type of modalities they produce for each application, the environment in which those applications are developed, and the eventual research applications those datasets are suited for based on the type of data modalities they produce.

Android Datasets. Android application datasets are assembled from the F-Droid repository, by Krutz *et al.* [48] and Pecorelli *et al.* [66] and from the Google Play Store, by Chand *et al.* [25], for various purposes, which we highlight.

Krutz *et al.* [48] collected metadata, version control information, source code, and the most recent APK file for 4,416 versions of 1,179 open-source Android applications from the F-Droid repository. Static analysis was conducted using three tools: Androrisk and Stowaway to analyze the APK files and Sonar to examine the extracted source code. The results of their research provide easy access to data and analytical results for software engineering research and a benchmark

dataset for comparison with other homemade static analysis research projects [48]. However, the modalities extracted are not suitable for security applications.

Pecorelli *et al.* [66] collected 1,780 Android applications by mining F-Droid to assess whether these apps are tested, how well they are designed, and how effective they are. A key finding of the study was that developers of mobile apps still tend not to test them properly. They also found that the test cases of the considered apps are of poor design quality, as measured by test code metrics and test smells, and their effectiveness, as measured by code coverage and assertion density [66]. The work is limited to testing and includes no additional modalities for other analysis.

Chand *et al.* [25] used Parsehub to scrape 394 applications from the Google Play Store with their metadata and a total of 8,423 user reviews to identify trends [25]. While valuable, the dataset is limited to gaming APKs and does not consider the range of possible application genres that could benefit from inclusion in a benchmark dataset. Moreover, the data does not include any data modalities that could benefit software and security analysis tasks.

Android Security. We sample three studies highlighting security analysis of Android applications. Two studies collected applications into datasets for Android security analysis [37, 77], and one tested Android devices [79].

Sun*et al.* [77] proposed a semantic-aware approach to identify and classify Android malware in large-scale datasets. Their system extracts API calls, permissions from APK files, and categories and descriptions from a mobile store. They applied Doc2Vec to represent these features semantically. The results showed a high F-measure of 99.71% and a low False Positive Rate (FPR) of 0.37% when identifying large-scale Android malware, and the highest F-measure of 97.5% in Android malware family classification [77].

Sun *et al.* [77] had three datasets with varying sample sizes. They assigned the first dataset as a baseline, the second as a large-scale Android malware identification, and the third as an Android malware family classification. For the first dataset, they developed a crawler to download 87,182 APK files from the Opera Mobile Store, scanned them through VirusTotal, and analyzed only 61,730 APK files. In the second dataset, they used the same crawler to collect APK files from the Google Play Store, scanned them through VirusTotal, and added 24,461 Android malware files. The last dataset was the same malware files from dataset 2 plus 72,227 (*i.e.,* divided into three groups 24,389, 24,157, and 23,681) Android malware files [77]. Even though this study administered a large-scale dataset and some applications were collected from the Google Play store, the dataset was analyzed on one aspect of security analysis: Android malware detection and classification. Felt *et al.* [37] developed Stowaway, which detects overprivilege in Android apps. Despite being seminal, the work is limited in data modalities and their security and software analysis applications.

A graph-based analysis of three datasets was performed by Alasmary *et al.* [13] to assess the similarities between IoT and Android malware and benign software for detection. Their data set

consisted of 2,874 IoT and 2,891 Android malware samples, delivering 97.9 accuracy in malware detection, although the analysis modality contributed is only graph-based and the application is only malware detection and code similarity analysis.

2.2 Privacy Policies in AR/VR

Privacy Policy Analysis. Privacy policies inform users about data collection, use, and protection practices. Numerous studies have focused on analyzing these policies. For instance, Alabduljabbar *et al.* [10] conducted a comprehensive analysis of privacy policies using a BERT-based technique, categorizing segments into predefined categories and showing trends in the analyzed policies for the presence or absence of various collection, use, and protection. Wilson *et al.* [85] and Andow *et al.* [15] developed tools for extracting and analyzing privacy policies to identify potential misalignment between the stated and the actual practices. These studies emphasize the need for advanced NLP techniques to help better understand and classify privacy policies. However, this analysis is done mostly for websites and does not consider privacy policies in AR/VR apps. Yu *et al.* [94] introduced PPChecker, a system that uses NLP and program analysis to identify issues in privacy policies: incompleteness, incorrectness, and inconsistency.

NLP Techniques in Privacy Analysis. NLP techniques, e.g., BERT, have advanced text classification and analysis. Devlin *et al.* [34] introduced BERT, a model that has revolutionized NLP by providing a deep contextual understanding of text. This model has been utilized in various domains, including privacy policy analysis, due to its ability to capture nuanced meanings and relationships within text. Elluri *et al.* [36] demonstrated BERT's effectiveness in enhancing text analysis and classification and showed significant improvements in accuracy and robustness, making BERT an ideal choice for analyzing complex privacy policy documents.

AR/VR Privacy Policies. Although studies by Harborth *et al.* [41] and Lim *et al.* [53] highlight the critical importance of transparent and comprehensive privacy policies in this domain, quantifying the gap in the practice from that of the expected privacy policy structure and compliance is still lacking.

2.3 Malware Detection Techniques

Malware Detection Using API. The use of APIs in malware detection remains a fundamental approach, especially in analyzing how malware interacts with system components. Xu *et al.* [86] proposed a model leveraging the BERT transformer and knowledge distillation to capture API sequences for malware detection. This approach significantly reduced training time while maintaining high detection accuracy, making it an efficient solution for dynamic environments. Tomiyama *et al.* [80] also explored dynamic analysis by focusing on API call sequences combined with mem-

ory dump analysis, resulting in an accuracy of 82.7%, highlighting the importance of runtime behavior in detecting advanced malware.

Moreover, RF models have consistently demonstrated their utility in malware detection, particularly for API-based analysis. Zhu *et al.* [99] employed RF combined with sensitive API monitoring, achieving an accuracy of 89.91%, showcasing the model's robustness for Android malware detection. Furthermore, recent developments have explored the potential of GNNs in API call graph analysis, where GNNs model the complex relationships between API interactions, providing deeper insights into malicious behavior.

Android Malware Detection. Given the popularity and relative vulnerabilities of the Android platform, a large body of research is presented in the literature focusing on Android malware detection. For instance, Yang *et al.* [88] proposed an advanced approach combining API clustering with NLP and machine learning techniques. Evaluated on a dataset of 42,450 malicious applications, their method achieved an F1 measure of 82.6%, demonstrating its effectiveness in detecting Android malware [88]. Similarly, Zhu *et al.* [99] used RF with sensitive API monitoring to reach an accuracy of 89.91%, emphasizing the effectiveness of API-based analysis.

Guyton *et al.* [40] presented a feature selection approach focusing on permissions, intents, and APIs for Android malware detection. Their method applied to a dataset of 119,000 applications, showed that combining these features improved accuracy compared to using them individually [40]. Additionally, Hou *et al.* [45] enhanced RF techniques for detecting Android malware, improving detection accuracy on a dataset of 1,536 applications. These studies underscore the role of machine learning models, API analysis, and feature selection in enhancing Android malware detection performance.

ELF and IoT Malware Detection. While much of the research focus remains on Android malware detection, research on ELF and IoT malware detection is gaining traction due to the proliferation of Linux-based systems and smart devices. Atitallah *et al.* [19] proposed an IoT malware detection method using Convolutional Neural Networks (CNNs) combined with RF voting. They achieve an impressive accuracy of 98.68% on the MaleVis dataset, which includes over 14,000 samples. This highlights the increasing reliance on machine learning models in securing IoT.

Zhou *et al.* [98] introduced APInspector, a hybrid malware detection method for ELF systems. By analyzing API embeddings and parameter sequences using a combination of Hierarchical Attention Networks (HAN) and Multi-Layer Perceptrons (MLP), their method outperformed traditional models in detecting malicious programs from a dataset of over 10,000 samples [98]. Sundarkumar *et al.* [78] presented a static analysis approach for IoT malware, providing critical insights into malware characteristics and behavior. This work demonstrates the growing importance of API and system call analysis in malware detection across different platforms.

Anwar *et al.* [16] contributed to the IoT malware detection field by introducing a static analysis method that relies on RF to characterize malware behavior, applying this approach to over 10,000

IoT samples. Their results demonstrated that static analysis, combined with RF, can effectively identify patterns in IoT malware [16]. These studies collectively illustrate the growing importance of detecting malware across different platforms, including IoT and ELF, where dynamic analysis and deep learning have proven highly effective.

3 x-droid: A Benchmark Dataset for AR/VR and Security Applications

3.1 Summary of Completed Work

This study focused on addressing the gap in benchmark datasets for AR/VR applications on mobile platforms. We successfully compiled a dataset of 408 AR/VR applications from the Google Play Store. We standardized multiple data modalities including control flow graphs, strings, functions, permissions, API calls, hexdump, and metadata. This dataset is poised to serve as a foundational resource for researchers aiming to enhance AR/VR applications' security and performance. Our work highlights the diversity of AR/VR applications and sets a new standard for security analysis. The dataset has been utilized to explore various research applications, demonstrating its utility and relevance to current AR/VR technology research.

3.2 Introduction

Extended Reality (XR) encompasses various interactive forms between humans and machines in real and virtual settings [17]. Extensive reality is defined by three forms: Augmented Reality (AR) [93], Virtual Reality (VR) [24], and Mixed Reality (MR) [76]. VR can collect explicit input and non-verbal data, including user gestures, physiological measurements, and how they use technology through different sensors [38]. AR is a technology that overlays digital information, such as images, videos, or 3D models, onto the real world [35]. MR is a technology that combines both VR and AR, allowing for real-time interaction and coexistence between physical and digital objects [32]. Using XR technology, a digital twin world simulates or mirrors the physical world, allowing users to interact with the virtual world [76]. With the help of advances in software, hardware, and algorithms, coupled with sensing in wearable devices, VR allows realistic physical interactions through computer-generated simulations in 3D environments [38].

AR/VR applications are growing and widely used for platforms beyond gaming and entertainment. This growth could be attributed to improving gadgets designed for AR/VR [18] and software development. Software is the driving force behind the AR/VR hardware, and there is an interest in understanding this software and its intentions, specifically from a security perspective. Despite the existing work in the literature for understanding Android software, there is scarce research addressing Android software targeting AR/VR deployment environments. Furthermore, given the continuous and substantial expansion of these types of applications and their widespread prevalence, it is crucial to assess any potential privacy and security issues that may arise [38] in addition to software analysis.

Limited datasets exist for applications that can be used as software and security analysis bench-

marks. The existing datasets are all related to applications targeting a variety of fields and users, none of which are solely relevant to AR/VR applications. For example, Nusrat *et al.* [64] mentioned a lack of studies and research on VR and VR software development in all platforms, including Android. The lack of studies can also apply to AR and AR software development.

Android software targeting AR/VR environments differs from Android software targeting mobile devices due to several distinct features, such as the AR/VR app's immersive nature, hardware requirements, performance demands, and unique interaction models. 1. A VR/AR app creates a virtual environment or enhances a real-world environment to create an immersive user experience, and these apps require different design principles and user interaction models. This affects the user interface design since AR/VR apps rely on immersive menus, spatial interface elements, gaze-based interactions, and gesture recognition. 2. AR/VR apps require specialized hardware, such as VR headsets with built-in displays and motion-tracking capabilities or AR glasses with cameras and sensors. The tracking of user movements and interactions depends on these sensors and different input methods in AR/VR apps. 3. AR/VR apps require considerable processing power and graphics capabilities (e.g., CPU, GPU, and memory). To meet these demands, it is necessary to create 3D environments, models, animations, recognize objects, map spatially, and handle occlusions. Therefore, AR/VR app developers should optimize their applications to provide smooth and responsive experiences. This is especially critical when considering the high frame rates required for VR applications to prevent motion sickness. There is limited literature that explores AR/VR Android dataset extraction, curation, and generation systematically that can be used for research tasks, and we take on this particular task to create this dataset.

3.3 Data Curation

This section provides information about the data, the collection and preparation processes as shown in Figure 1, and the tools used to acquire and analyze the dataset. As a starting point, a few background details (e.g.,, the Google Play Store, APK, SerpApi, emulator, and VirusTotal) were presented. Additionally, the process of collecting and preparing data was discussed. As a final point, five tools have been described and highlighted.

3.3.1 Background

Google Play. As Google's main store, it offers a range of contents, including applications, movies, TV shows, e-books, etc. The Google Play store was created by merging the Android Market, the Google e-book store, and the Google Music store. Users can access Android applications, known as Android Package Kits (APKs), available for Android devices in this store. In addition to Android, Google Play supports other operating systems (OS) that can access certain store features, including Windows. Google Play supports multiple platforms, including Windows, smartphones,

tablets, TVs, Chromebooks, and watches.

APKs and apps have webpages that provide information the apps, such as its name, developer, rating, download number, and thumbnail. Information on applications is categorized into numerical and non-numerical categories. Among the numerical categories are ratings, downloads, reviews, and dates. Nonnumerical categories include the name of the APK file, the developer, the description, comments, data safety information, and compatible platforms.

Among the categories available on Google Play that are not numerical are age-based ratings. Using this method, the minimum maturity level of the content is determined for each application. Ratings are based on countries and regions and are divided into seven categories. Based on the fact that the sample was collected in the United States and that the rating system was derived from the Google Play store in the United States, we have chosen a rating system for North and South America. According to the Entertainment Software Rating Board (ESRB), the age-based rating for the North and South American regions consists of categories for everyone: everyone + 10, teens, mature, and adults [1]. Without a rating for age, applications are considered to have highly mature content for parental control.

APKs. For the Android OS, applications are contained in a file format called the APK. Because APK files are limited to about 100 megabytes, some applications may have an Opaque Binary Blob (OBB) expansion file. Android devices are the only platforms compatible with these APK files, which cannot be used in other systems. Moreover, because the user will have access to the entire file (APK or OBB), this paper considers every application to be a single APK file, irrespective of the impact of OBB files.

Java is the most widely used programming language for developing Android applications. It was the primary language used in the development and is still widely used. Android Studio, the official integrated development environment (IDE) for Android development, supports Java.

Google SERP API. Developers can automatically extract information from Google search engine results using the Search Engine Results Page (SERP) Application Programming Interface (API). APIs such as SerpApi, SerpStack, or scrape-it work by sending a query to third-party endpoints. One of these tools is utilized to extract data from the Google Play store to minimize Google Play's search history, device, location, and many other factors that could limit or hinder the results. Utilizing a SERP API tool to scrape results pages will allow access to results without personalized filters and prioritization. SerpApi was utilized to collect metadata for our dataset, which provides all responses in JSON format and contains the relevant details, such as titles, link titles, descriptions, product IDs, ratings, reviews, downloads, and any other relevant details. Many Google API calls are supported by this search tool, including the Google Play store, Maps, Images, Shopping, etc.

SerpApi supports Python, Java, PHP, Ruby, and Node.js, among others. As a result, developers can seamlessly integrate SerpApi into their existing workflows and projects. Through SerpApi,

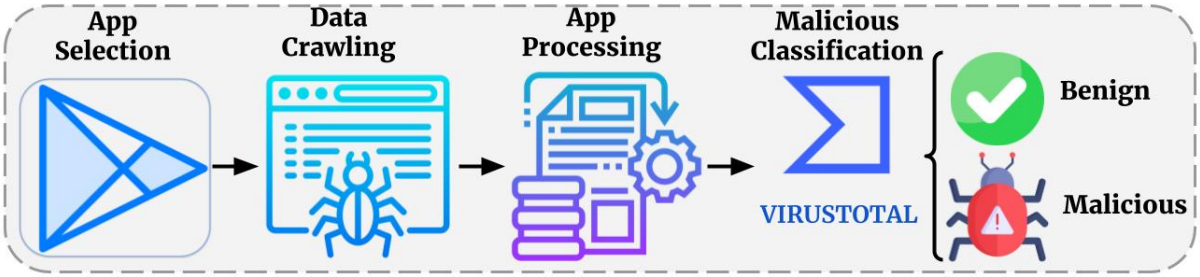

Figure 1: Dataset Collection Pipeline

users can send requests to search engines such as Google in real-time, enabling them to receive fast results. Additionally, structured search results are available in a programmable format, JSON. It allows viewing of SERPs from designated locations through global proxy servers. It can bypass CAPTCHAs and retrieve structured data accurately and timely. In this tool, researchers can specify search parameters and receive detailed information on search results; however, there are some limitations associated with its implementation. There is a limited selection of output formats supported by SerpApi; it cannot perform searches on specific devices, and on average, its search speed is slower than other Google SERP API tools.

Emulator. Samples were collected directly from the Google Play store using an emulator and transferred to a different OS to analyze the applications further. The Galaxy S20 Ultra phone, a device suitable for acquiring and running these files, was emulated. Moreover, the Google Play store allows developers to label their apps according to the platform for which they develop their software (e.g., smartphone or tablet). The study platform was an Android phone, so most reviews and ratings were based on the Google Play store for phones. Exceptions to this rule are applications that do not have a label. Among the sampled applications are those designed for mobile phones only, tablets only, smartphones and tablets, or unspecified.

VirusTotal. As an online malware detection platform, VirusTotal offers more than 60 detection tools for single-file searches. Several methods are available for the platform to examine a file, including uploading it or looking at the hash value associated with each file. It could take various factors to determine the estimated turnaround time for the results, including file size, queue length, and scanning method. However, if the hash value of the file had already been stored in the database, it would take less time to scan the same file in the future. A hash value is used to identify the file on the platform. The system provides detailed responses that include the date and time of the scan, the scan result, and the analysis tools used to determine the final result.

3.3.2 Dataset Collection Pipeline

The samples were collected by searching for four terms within the Google Play platform using SerpAPI: VR, virtual reality, AR, and augmented reality. More than 430 applications were found in the search result, but only 408 were available to download and compatible with the emulator.

A few applications are unavailable for download because there is no software available on the application website or because of compatibility issues with the platform. As described above, these 408 applications serve as the sample for the study that was administered to extract and collect all of the dataset's characteristics. These applications ranged from less than 100 KB to more than 1.5 GB.

The*BlueStacks* software was used to emulate the phone and collect APK files within the Android OS. To access these files through Linux and perform the analysis, the files were uploaded to private Google Drive as APK files. Using an emulator ensures that the phone is the primary platform for the application. Consequently, it would be possible to prevent encountering incompatible applications with the primary platform. Using an emulator is also beneficial because it provides direct access to the Google Play store. As such, there is no possibility of encountering third parties that could significantly impact the dataset or the results. In addition, *EX File Manager: File Explorer* was used to extract and upload APK files from the Android system to Google Drive for further analysis. To analyze the dataset and extract the data characteristics presented in section 3.4, we downloaded the APK files to the Linux OS.

Subsequently, VirusTotal examined it to detect malicious data within each app and assign labels. The findings of the VirusTotal scan are discussed in more detail in section 3.4.

3.3.3 Analysis Tools

As illustrated in the data application section, five tools have been employed to examine the dataset and extract additional data, see Figure 2 and Table 1. Each tool has been used with specific parameters and for particular tasks, ensuring a comprehensive understanding of the dataset components. The details provided aim to assist researchers in replicating the study or applying the methodology to new or private applications.

Listing 1: Radare2 aaa steps

```
[0x00000000]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Constructing a function name for fcn.* and sym.func.* functions (aan)
[x] Type matching analysis for all functions (afta)
[x] Use -AA or aaaa to perform additional experimental analysis.
```

Radare2. Radare2 is an open-source framework for reverse engineering and binary analysis. It is extensively used by security experts and software engineers for disassembling code and debugging programs. In our research, Radare2 was employed to analyze binary files from the APK datasets.

Usage Details. Radare2 is employed to disassemble and analyze binaries from the APK datasets. Extracted modalities are functions, strings, API calls, and hexdumps, see Figure 2.

Parameters and Time. We used the ‘aaa‘ command for extensive analysis, taking about 2 minutes per APK, depending on complexity and size. Other commands may require more time depending on the specific analysis performed.

Error Handling. No significant errors were encountered when using Radare2, ensuring a reliable dataset for analysis.

APKtool. APKtool is utilized for decompiling and recompiling APK files, allowing for a deeper inspection of Android applications. APKtool is pivotal in extracting the manifest files and other resources from APKs, which are critical for understanding permissions and configurations.

Usage Details. APKtool is used for decompiling APK files to extract control flow graphs, as shown in Figure 2, and other resources, critical for understanding configurations.

Parameters and Time. APKtool decomposes APK files to retrieve XML representations of resources, typically taking about 1 minute per APK, depending on the file size.

Error Handling. APKtool consistently provides accurate decompilations without any major errors to report.

Jadx. Jadx converts APK files into readable Java source code, assisting in the understanding of its logic and functionality.

Usage Details. Jadx is used to convert APK files into readable Java source code, aiding in the extraction of control flow graphs and the understanding of application logic as demonstrated in Figure 2.

Parameters and Time. Jadx with default settings, including cross-referencing, took at least 3 minutes per application.

Error Handling. Jadx performed reliably without errors, contributing to effective source code analysis.

Fernflower. Developed by JetBrains, Fernflower is used for decompiling Java bytecode into readable Java source code.

Usage Details. Fernflower is used for decompiling Java bytecode back into readable Java source code, mainly for analyzing Java-based applications.

Parameters and Time. Standard decompilation process, taking about 2-3 minutes per Java class file.

Error Handling. No significant issues were noted, ensuring the integrity of the decompiled code.

AAPT2. Android Asset Packaging Tool 2 (AAPT2) compiles and packages Android resources, used in our study to analyze how resources are managed and optimized in APK files. The AAPT2 was implemented to extract permissions from the APK files.

Usage Details. As illustrated in Figure 2, AAPT2 compiles and packages Android resources and is utilized to analyze API calls and how resources are managed within APK files.

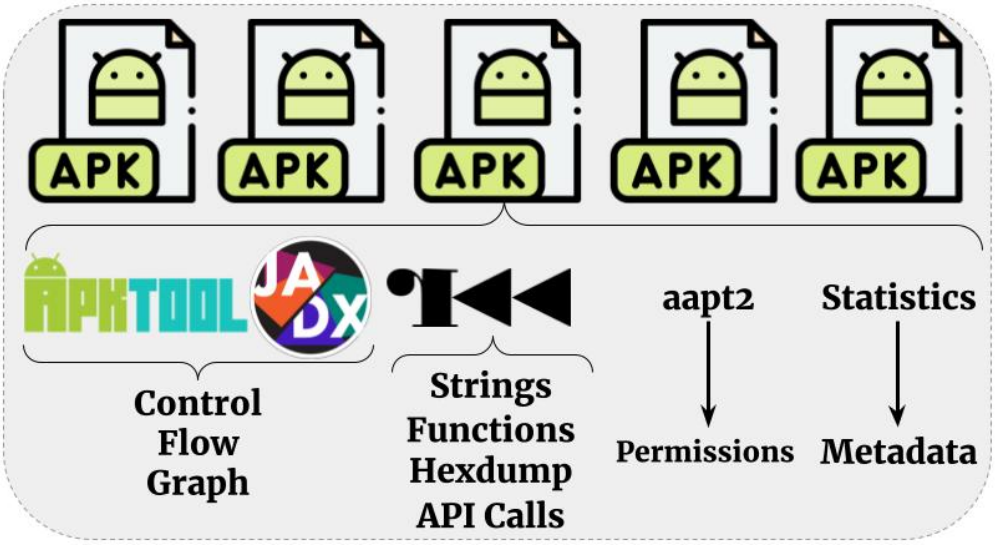

Figure 2: Dataset features.

Table 1: APK Features and Analysis Tools

Tasks	File Type	Analysis Tool	Program Code
CFG	dot	APKtool& Jadx	Python
Functions	csv	Radare2	Shell script
Strings	csv	Radare2	Python
Permissions	csv	AAPT2	Shell script
API calls	csv	Radare2	Shell script
hexdump	txt	Radare2	Shell script
Metadata	csv	Statistical Analysis	None

Parameters and Time. AAPT2 handles resource compilation and linking, completing in under 30 seconds per APK.

Error Handling. AAPT2 provided consistent and error-free outputs during resource compilation.

3.4 Data Modalities and Applications

We highlight all results, graphs, and tables related to all features. In addition, each feature was subdivided into subsections. Seven subsections have been discussed: control flow graphs, APK functions, APK strings, APK permissions, APK API calls, APK hexdumps, and metadata. The information in each subsection provides a concise introduction to each feature and comprehensive information about the output files. To illustrate the practical application and relevance of these modalities, we present a detailed case study on the security evaluation of selected AR/VR applications.

3.4.1 Control Flow Graph (CFG)

CFGs represent control flow within a program [3], from one statement to another and are represented graphically in this graph. In various stages of software analysis, including compilers, code analysis, and reverse engineering, CFGs are used [87, 92]. CFGs can provide valuable insight into a program's structure and performance, identify potential optimization opportunities, detect unreachable code, and perform various static analyses [5], such as data flow analysis and reaching definitions analysis [62, 96]. Likewise, they are employed in reverse engineering to assist in visualizing compiled code's control flow. As a result, binary programs may be understood without access to their source code.

A CFG comprises key components: basic blocks, nodes, edges, and entry and exit nodes. In a basic block, the control flow enters at the beginning and exits at the end of a series of consecutive statements [11]. The flow is uninterrupted, with no stops or breaks except at the end. Therefore, CFGs are made up of basic blocks. Each node in a CFG represents a basic block [3], and nodes represent specific points where control can be entered or exited. A CFG's edges illustrate the control flow between the basic blocks [92]. The edge between two nodes signifies a possible transfer of control from one basic block to another. An entry node is the point at which the CFG begins [11]. In other words, it represents the start of the execution of a program. The exit nodes indicate that the execution of a program has come to an end [92]. Return statements or exit functions can be used to exit the control flow of a program at various points.

Research Usage. Studies have been conducted in the literature examining CFGs and their functions for analyzing and comprehending AR/VR applications. In addition to providing researchers with a dataset, we will allow them to analyze the following aspects related to AR/VR applications, which will complement the findings of their studies.

① *Malware detection and analysis.* By analyzing the control flow of APK files, malware can be detected [4], or harmful code patterns can be identified [96]. For example, when detecting malware, nodes count, edges count, or other CFG features [13] could be considered. To detect anomalies or suspicious patterns in a control flow, researchers can automate the generation of CFGs from APKs using machine learning [11, 44] or pattern recognition algorithms [3, 12].

② *Security analysis.* CFG analysis can identify security vulnerabilities in apps [12], such as insecure data storage, incorrect input validation [87], and weak authentication. The control flow can be analyzed to identify sensitive data paths or critical system resources and assess security implications.

③ *Software verification.* CFGs is used to improve Android app testing techniques [87]. To achieve better code coverage and detect defects or vulnerabilities more effectively, researchers can develop automated testing tools that generate test cases based on the control flow structure of the APK.

④ *Behavioral analysis.* In addition to providing insight into Android apps' runtime behavior [87],

CFGs can also provide insight into their design. Analysis of the control flow can reveal performance bottlenecks [44], identify resource usage patterns, or identify how different components interact.

⑤ *Code understanding and optimization.* CFG analysis can help optimize the performance of the Android app [3, 87] or understand its code structure. CFGs can be visualized, redundant code or inefficient algorithms can be identified, and optimizations can be suggested to improve the overall efficiency and maintainability of the code [87].

3.4.2 Functions

An Android application comprises multiple components and functions, each contributing to its overall behavior and features. Each function file delivers detailed information about all the functions included in the respective APK file. The output for each APK is a CSV file that lists the name, length, and additional details about the functions.

Research Usage. In the literature, various studies have examined functions to understand and analyze the behavior of the applications. Researchers can use our dataset to complement their study of AR/VR applications by gaining a better understanding of the following aspects.

① *Function Call Graph (FCG) analysis.* FCGs are constructed by analyzing APK files to identify the functions and methods present in the code and their relationships [75]. This graph represents the control flows between functions and can be used for various purposes, such as understanding the code structure, identifying dependencies, and detecting potential vulnerabilities [58].

② *Code Smell Detection.* Function-level code smells in Android applications can be analyzed [67, 97] to understand defects and maintenance difficulties that indicate poor design or implementation processes [97]. Some of the insights that can be uncovered through function-level code smell analysis include methods that are too long [67], excessive parameter lists, inconsistent naming conventions, and potential remedies; e.g., refactoring [46, 97].

3.4.3 Strings

An APK string refers to a string used within an Android app and includes text from the user interface, labels, and other textual elements. As part of the app's resources, strings.xml contains these strings. The strings.xml file can be found in the res/values directory of the Android app's project structure. The file contains string resources used throughout the application to display text to the user. Strings defined in strings.xml are accessed from the Java or Kotlin code of the application using the getString method [82], allowing to reference text resources dynamically and flexibly.

Research Usage. The literature contains many studies that analyze string behavior to understand and analyze applications' behavior. Using our dataset, researchers can better understand the fol-

lowing aspects of AR/VR applications.

① *String Analysis for Localization.* String analysis within APK files can be used to identify language [63] and regional-specific text [29] that needs to be localized. In this process, strings are extracted from the application's resources and categorized according to their context, and recommendations are given to translators for effective localization [29].

② *Analysis for Semantic Understanding.* APK files include strings describing the app's functionality, user interface, and behavior [51, 82]. By analyzing and parsing strings [28, 33], researchers can infer user intentions, identify key features of an application, or extract semantic meaning.

③ *String Encryption and Obfuscation.* Reverse engineering and tampering can occur with string literals within APK files [33, 63]. Using obfuscation or encryption within the application code [33, 63], researchers can protect sensitive information, such as API keys or cryptographic keys [63], from attackers easily extracting it.

④ *Malware Detection and Analysis.* Strings in APK files can contain indicators of malicious behavior [30, 33, 63, 82]; e.g., hardcoded command-and-control server addresses [16], obfuscated URLs associated with payloads [82], or suspicious API calls [63, 82]. With string analysis techniques, Android applications can be analyzed to detect malware and potentially unwanted behaviors [29, 30].

⑤ *Privacy Analysis.* There is a potential for sensitive information to be inadvertently exposed within APK files, e.g., personally identifiable information, authentication tokens, and identifiers [33]. Android developers can identify and remediate security and privacy vulnerabilities by developing techniques to analyze strings and detect privacy violations.

3.4.4 Permissions

APKs in Android include a manifest file (AndroidManifest.xml) that contains essential information about the app [21, 27]. Defining the permissions for an app is an important aspect of its manifest file [61]. To access certain features or data on an Android device, applications require a variety of permissions. These permissions are declared in the manifest [47] and must be granted by the user when installing the app. When downloading an application from the Google Play Store, the user is presented with the permissions it requests. The user can review the permissions displayed before downloading the app, and they can decide whether to accept or deny the installation process. Furthermore, users can revoke specific permissions for applications installed on their device through the settings menu.

Permissions can fall into several categories, including normal permissions [27, 83], dangerous permissions [57, 61, 69], and signature permissions [37, 52, 69]. Normal permissions are considered harmless [37] in terms of user privacy. Users are automatically granted [52] when they install the app. Examples include accessing the Internet or accessing the network state. Dangerous per-

missions are sensitive and can compromise user privacy or data security [61]. Examples include accessing the device's camera [21], contacts, location [69], or reading SMS messages [52]. For dangerous permissions, the user must explicitly grant permission during run-time on Android devices [61] running Android 6.0 (Marshmallow) or higher [83]. Applications targeting Android 5.1 (API level 22) or lower request dangerous permissions at the time of installation. Signature permissions are granted only if the application requesting is signed with the same certificate as the application that declared the permission [37]. This mechanism is used to communicate between processes and share data between applications from the same developer.

Research Usage. There have been multiple studies in the literature that explore the use of mobile application permissions to analyze and comprehend those applications and their functions. Our dataset will allow researchers to complement their studies by understanding such aspects in the context of AR/VR applications, including the following.

① *Permission Analysis and Classification.* Analyzing the permissions requested by Android apps allows us to understand their purpose and potential risks [77]. Studying the relationship between requested permissions and an application's functionality involves categorizing permissions and categorizing them according to their sensitivity (e.g.,, camera access, location access, contacts access).

② *Permission Misuse Detection.* Detecting when permissions are misused or excessively misused by Android apps can be done [69] by identifying unnecessary permission requests [37], patterns of permission abuse [69], or discrepancies between declared and used permissions [83].

③ *Privacy Risk Assessment.* Android apps may be granted access to sensitive resources on a device through lax permissions, which this may pose a privacy risk [52, 57, 61, 83]. Using privacy controls and consent mechanisms, researchers can assess how requested permissions impact privacy, identify sensitive data exposed to third-party applications, and assess the adequacy of privacy controls.

④ *Behavioral Analysis and Permission Usage.* During the execution of Android applications, researchers can analyze how permissions are used during the run-time process [61]. An example would be to monitor permission usage patterns [21, 37, 69], following interactions with sensitive data, and identify privacy violations that could result from improper permission handling [69, 83].

⑤ *Users Perception and Trust.* Researchers can investigate how Android users perceive and understand permission requests. It may also involve exploring strategies to improve transparency, communication, and trustworthiness in permission requests by studying user attitudes about permission requests and factors influencing users' decisions.

Table 2: Overview of metadata

Feature	Description
Name	The application official name
Link	The application link
ID	Unique identifier
Version	The application version at collection
Developer	Developer information; e.g., name
Extensions	In-app purchases, ads, or both
Rating	Five-star rating value
Reviews	The reviews for the application
Age	The age group per ESRB
Downloads	The number of downloads
Category	The application genre.
Updated	The day, month, and year of app update
Platform	Compatible Platforms
Safety	Safety practices provided by the developer

3.4.5 API Calls

API calls are methods a software application provides for other programs. In Android apps, API calls interact with the Android OS, hardware, or external services [88]. API calls to APK (Android Package) files are typically related to the interaction between an Android application and external services, libraries, or resources. API calls are crucial in enabling communication between different components of an Android application and external entities [49]. Android provides a set of APIs that developers can use to access various features and functionalities. This includes APIs for user interface components, data storage, networking, and more. Developers can make API calls to take advantage of these features and integrate them into their APK files.

Many apps make API calls to external servers to fetch and send data or perform network-related tasks [37]. This is commonly done using libraries such as Retrofit or Volley. These calls are crucial for applications that require real-time updates. Android apps must declare permissions in their manifest file to access certain features or perform specific actions, such as accessing the internet or using the device's camera [77]. API calls that require these permissions must be handled appropriately to ensure security and user privacy. Secure API calls are essential to protect user data and ensure the app's integrity. This involves using secure communication protocols (such as HTTPS), properly handling authentication (e.g., OAuth tokens), and implementing secure coding practices to prevent common vulnerabilities like SQL injection or Cross-Site Scripting (XSS).

Android developers often use third-party libraries and SDKs (Software Development Kits) to simplify making API calls [84]. These libraries encapsulate the complexity of handling network requests, parsing responses [37, 57], and managing asynchronous tasks. Proper testing and debugging are crucial when working with API calls in Android applications. Developers use tools like Android Studio's built-in debugger and network inspection tools to identify and fix issues related

to API communication [52, 57]. Thus, API calls for APK files in Android development are essential for enabling communication between the application and external services or resources. This interaction is a key aspect in building feature-rich and dynamic Android applications.

Research Usage. API calls have been explored in the literature to understand application behavior. In addition to understanding AR/VR applications, researchers can use our dataset to complement their studies in the following areas.

① *API Dependency.* Code analysis involves detecting API calls to system APIs, third-party libraries, or cloud services and understanding their dependencies and patterns [84].

② *API Misuse Detection.* Techniques can be developed to detect misuse [20] and improper use of APIs within Android apps [55, 84]. Identifying potential vulnerabilities, performance issues, or compliance violations due to incorrect API usage may require analyzing API call sequences [37], parameter values [88], or error handling mechanisms [95].

③ *Security Analysis.* There are several security risks pertaining to API calls within Android apps, such as data leakage, bypassing authentication, or unauthorized access to sensitive data. Analysis of the interaction between the app and external APIs can identify security vulnerabilities, assess the potential impact of abused APIs, and develop mitigation techniques.

④ *Privacy Assessment of API Usage.* Applications use APIs to perform their functions, requiring permissions or access to sensitive data and raising privacy concerns regarding how sensitive information is handled. Using API calls, researchers can analyze privacy implications [49], identify potential data leakage risks, and assess the adequacy of the privacy controls and consent mechanisms in place.

⑤ *Behavioral Profiling and Anomaly Detection.* Using API calls, it is possible to figure out the running behavior of Android applications [73], including their communication patterns, resource utilization, and external dependencies [59]. Analyzing API call traces can help researchers profile applications' behavior, detect anomalies or deviations from expected patterns, and identify potential security and performance problems. To understand API adoption, popularity, and usage patterns within Android applications, researchers can analyze API usage patterns [49, 88] within the applications. Analyzing large datasets of APK files is one method to identify common API usage patterns, understand how API usage evolves, and identify emerging development trends.

3.4.6 Hexdump

A hexdump is a hexadecimal representation of the binary data in a file or memory region [2]. It is commonly used to debug, reverse engineer, and analyze binary files. In a hexdump, each byte of the binary data is represented by two hexadecimal digits [101] (from 00 to FF), providing a human-readable view of the data's raw contents. In addition to the hexadecimal representation, a hexdump often includes an ASCII representation of the data [2]. Non-printable characters (those

outside the ASCII printable range) are typically represented as dots or other placeholders.

Research Usage. Various studies have explored hexdump analysis to understand the internal structure, identify patterns, and detect anomalies in APK files. By gaining a deeper understanding of these aspects, researchers can complement their study of AR/VR applications with our dataset.

① *Binary analysis.* Hexdump can be used to analyze the binary structure of APK files [31], including headers, metadata, and executable code segments [101]. A hexadecimal representation of a file can be used to identify the signature, extract metadata, such as package names and version numbers [72], and reverse engineer the application's binary code to determine how it behaves and functions.

② *File format analysis.* Researchers can analyze hexdump files to understand the APK file format, assets, and code compile structure. They can also analyze the relationships between components of an app package by extracting embedded resources such as images, audio, or XML files and analyzing the hexadecimal representation of individual APK files.

③ *Malware detection and analysis.* Using hexdump analysis, malware can be detected and analyzed within APK files. Researchers can inspect hexadecimal representations of binary codes to detect suspicious patterns [2]. For example, hidden commands, encrypted payloads [72], and obfuscated code [2, 72] can be found. The detection and mitigation of malware threats targeting Android applications can be improved by identifying anomalies or malicious signatures.

④ *Security analysis.* Security vulnerabilities in APK files can be identified by hexdump analysis, including insecure file permissions, hard-coded credentials, and buffer overflows. Hexadecimal representations of files are useful to researchers in identifying potential security risks and developing techniques to assess and mitigate those risks. This would improve the security posture of Android applications.

⑤ *Data privacy.* Data leakage risks or privacy violations can be detected within APK files using hexdump analysis. Researchers can identify sensitive data within apps by examining the hexadecimal representation of the files [22]. Protecting user privacy and preventing data breaches require researchers to develop techniques for analyzing and sanitizing sensitive data.

3.4.7 Metadata

This subsection consists of findings derived from statistical analysis of data of various categories extracted directly from the Google Play store pages of the application and the APK files based on the information in Table 2. The findings were discussed in five main aspects or factors: Prevalence of malware, safety, domain, audience, and platform. Average (*i.e.,* mean, median, and mode) and simple mathematical operations were performed to evaluate each aspect and provide the analysis.

Table 3 provides an overview of the dataset groups, including average values for ratings, downloads, and the most recent activity year. Each row represents a different group within the dataset:

Table 3: Average values for separate groups

Group	Rating	Downloads	Activity
Sample	3.48	≤50k	2023
Malicious	3.22	≤500k	2018 & 2021
Benign	3.5	≤50k	2023
Everyone	3.5	≤50k	2023
Everyone+10	3.77	≤500k	2023
Teens	3.35	≤500k	2022
Mature	3.38	≤500k	2021

"Sample" refers to the overall dataset, while "Malicious" and "Benign" indicate sample subsets categorized based on harmful content. The "Group" column categorizes the data into these subsets, with ratings averaged across all apps within each group, download figures approximated to the nearest significant count, and the activity indicating the most recent year in which the apps were actively updated.

In Table 3, "Group" is used to distinguish between overall samples and specific subsets based on security assessments (malicious or benign). It is important to note that the groups malicious and benign are strictly exclusive subsets, with no overlap. Each application is categorized strictly as either malicious or benign based on our security criteria, ensuring clear separation for analytical purposes.

Table 4 categorizes the applications into different segments based on their primary function or market, such as 'Education', 'Games', etc. This table uses the same column headings as Table 3 but applies them to these functional categories instead of the type of content (malicious vs. benign). Each 'Group' in this table refers to a functional category, providing insights into each category's average ratings, download counts, and recent activity.

Prevalence of Malware. The first factor is the prevalence of malware in Android VR/AR apps, which can be explored by examining the result of the VirusTotal scan. VirusTotal aggregates the detection results from multiple antivirus scanners to assess whether an application is malicious. For our dataset, an application is labeled as malicious if it meets the following criteria:

- **Detection Threshold:** An app is considered malicious if detected by at least one scanner. This threshold can vary depending on the specific security requirements.
- **Detection Range:** The number of scanners detecting apps as malware ranges from 1 to 5 in our study.
  - *Roller Coaster Sunset app* was flagged as adware by 3 different scanners: Avira, Ikarus, and Fortinet.
  - *VeeR app* was flagged by 5 scanners, including ESET-NOD32 and Microsoft, indicating a higher consensus among scanners about its threat.

This information helps in understanding the variability and reliability of malware detection across the dataset, providing insights into the security of the apps we analyzed.

Based on the scanned apps, 16 out of 408 were labeled malicious by VirusTotal, and three age groups were the targets of those applications. 68.75% of the malicious applications were developed for the Everyone age group. In contrast, 25% of the targeted population for the malicious apps were teens, and only 6.25% were designed for the mature age group. Moreover, 11 out of 16 were classified as games and entertainment, which explains why malicious AR/VR applications have a higher average download group than the entire sample and benign, per Table 3. Furthermore, more than half of the malicious AR/VR applications (*i.e.,* 10 out of 16) were last updated +3 years ago, which is approximately 62.5% of the malicious applications. Table 3 shows that malicious application activity was both in 2018 and 2021, which implies that these malicious applications are not periodically updated.

Safety Landscape. Based on the safety analysis of the sample data, confidentiality, availability, and integrity have been the lenses used to address the second factor of the current safety landscape for AR/VR apps. About 65% of the apps in the sample have stated their data safety in all three of their categories (e.g.,, sharing, collection, and safety practices). The results showed major concerns about data safety. Most of the data in the sample is not encrypted while transmitted, and only 39.46% of the data sample is encrypted.

Another issue is the availability and confidentiality of users' data. A small number of apps stated that they share their data with third parties, 67 out of 408, or 16.42%. Furthermore, very few applications authorize the user to delete their data in the applications. In contrast, the remaining applications do not authorize users for applications that state their data safety information or other applications that do not specify whether the user is authorized to do that. Only 25.25% of the sample size gave their users permission to control their information in the applications.

Application Domains. To address this factor, application categories and various groups have been examined, such as the age group applications in Table 3 and different categories applications as demonstrated in Table 4. The analysis of the application categories showed that education is the highest, with 91 out of 408, followed by games and entertainment, respectively; see Table 4. The increase in education is attributed to the increase in educational AR/VR applications in the last two years, 2022 and 2023. Approximately 62.64% of the applications that were classified as educational were created in 2022 and 2023, as shown in Figure 3.

Based on the analysis of the most recent and highest activity applications with those in 2023, VR/AR applications are shown more geared toward education and entertainment. Moreover, three categories that dominated applications in 2023 were education at 18.70%, entertainment at 15.45%, and tools at 12.20%, as highlighted in Figure 3. However, the AR/VR games started declining in the last two years, 2022 and 2023, and the line of decrease for games compared to education and entertainment can be clearly shown in Figure 3.

Table 4: Average values for AR/VR categories.

Group	Apps #	Rating	Downloads	Activity
Education	91	3.53	≤50k	2022
Games	66	3.33	≤500k	2021
Entertainment	57	3.38	≤50k	2023
Simulation	39	3.60	≤500k	2022
Tools	29	3.71	≤50k	2023
Video Players	19	3.61	≤500k	2022
Business	15	3.66	≤500k	2023
Art & Photo.	15	3.76	≤50k	2023
Casual	14	3.48	≤50k	2022
Books & News	14	3.90	≤50k	2022
Social & Comm.	11	3.40	≤50k	2023
Lifestyle	10	2.50	≤50k	2020
Productivity	8	3.67	≤50k	2023
Sports	7	2.96	≤50k	2020
Health	5	3.47	≤50k	2022
Shopping	4	1.8	≤50k	2019 & 2022
Travel & Maps	4	4.33	≤100k	2023

Targeted Audience. As mentioned in the previous section, the app platform divided users into six groups, and only four groups were targeted by the AR/VR apps: Everyone, Teens, Everyone +10, and Mature. The Google Play Store categorizes applications using an age-based rating system to help users identify appropriate content. The categories are defined as:

- **Everyone**: Suitable for all ages.
- **Everyone +10**: Suitable for children 10 and older. May contain more cartoon, fantasy or mild violence, mild language, and minimal suggestive themes.
- **Teen**: Suitable for ages 13 and older. It may contain violence, suggestive themes, crude humor, minimal blood, gambling, and infrequent use of strong language.
- **Mature**: Suitable for ages +17; includes intense violence, blood and gore, sexual content, and strong language.

These categories are instrumental in segregating the apps for our study, where ratings influence the grouping and analysis, e.g., Table 3. Furthermore, understanding the target audience of AR/VR applications is crucial for several aspects of software development and research:

- **App Design and Functionality:** The target audience significantly influences the design and functionality of applications. For example, apps designed for children might feature more interactive and graphical interfaces with safeguards against inappropriate content.

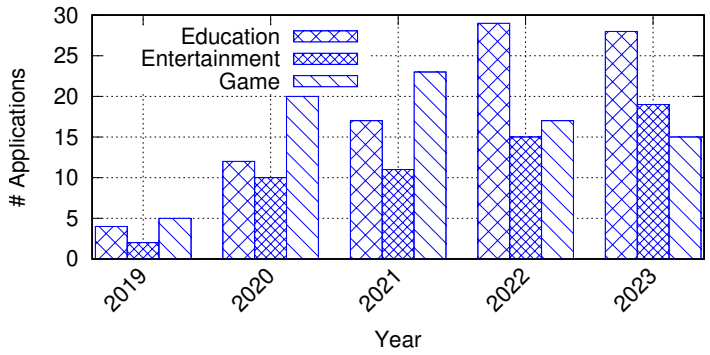

Figure 3: *Education*, *Entertainment*, and *Game* application numbers per year for the years 2019 through 2023.

Table 5: A summary of the modalities and use cases in *x-droid*, including Malware Detection (MalD), Security Analysis (SA), Software Verification & Validation (SVV), Software Optimization (SO), Behavior Profiling (BP), and Encryption & Obfuscation (EO), Localization (L), Semantic Understanding (SU), Privacy Analysis (PA), Analysis & Classification (AC), Misuse Detection (MisD), Users Perception (UP), Binary Analysis (BA), File Format Analysis (FFA), Anomaly Detection (AD). The average # represents the number of graphs, functions, strings, permission groups, API calls, and metadata groups on average per app. Hexdump is not reported as the number depends on the representation (Byte or word).

Modality	Average #	MalD	SA	SVV	SO	BP	EO	L	SU	PA	AC	MisD	UP	BA	FFA	AD
CFG	82,854	✓	✓	✓	✓	✓										
Functions	5,741	✓	✓	✓	✓	✓										
Strings	934,601	✓					✓	✓	✓	✓	✓					
Permissions	11					✓				✓	✓	✓	✓			
API calls	1,680		✓			✓				✓	✓	✓				✓
Hexdump	*	✓	✓							✓				✓	✓	✓
Metadata	5	✓	✓							✓	✓		✓			

- **Security and Privacy:** Different age groups necessitate varying levels of security and privacy protections. Knowing the target audience aids in tailoring security measures to protect sensitive user data appropriately, especially in applications designed for minors.

- **Regulatory Compliance:** Apps intended for specific age groups must comply with different regulations, such as those protecting children's online privacy. Analyzing the target audience and associated features helps ensure that apps meet these legal requirements.

In our dataset, apps that target teens often incorporate more social features and real-time communication capabilities, which require robust data protection measures to ensure privacy and compliance with youth protection laws. Thus, target audience analysis enhances our understanding of the app's design and security framework and provides a foundational element for comprehending broader usage patterns and regulatory alignment in AR/VR applications.

Most of the samples (82.84%) were labeled as *Everyone*, followed by the Teen group (11.03%), then Mature +17 and Everyone +10 at 3.19% and 2.94%, respectively.

Analyzing targeted audience applications, especially the Teens group, revealed that games and

Table 6: Top 5 permissions requested in *x-droid* against per-age group permissions highlighting trends and affinities.

#	Sample	Everyone	Everyone 10+	Teen	Mature
1	internet	internet	read_external_storage	internet	internet
2	access_network_state	camera	internet	access_network_state	provider_insert_badge
3	camera	access_network_state	billing	camera	write_settings
4	read_external_storage	write_external_storage	custom 1	read_external_storage	update_shortcut
5	write_external_storage	read_external_storage	custom 2	wake_lock	broadcast_badge

entertainment are the most dominant category for that group. Games and entertainment had the highest percentage of categories, with 75.56% in VR/AR apps for teens. This also explains why teenager applications have higher downloads than the entire sample, per Table 3. Only 1.1% of the educational apps developed for teens (*i.e.,* one educational application of the 91 apps for the entire sample). An educational app for teens, developed in 2015, indicates a lack of educational apps for teens, with the recent increase in the education category in the last two years. Furthermore, most of the teens' applications (*i.e.,* 60%) contain advertisements or have in-app purchase options. Therefore, the financial aspect is the driving force behind the creation of teen applications due to the high number of games and entertainment apps and the lack of educational apps.

Cross-Platform. Several platforms are available for downloading applications, including the phone and tablet. An examination of cross-platform capabilities prevalent in the most popular AR/VR applications (*i.e.,*, applications that have been downloaded a million times or more) and whether those applications are adopting this capability has been carried out. Furthermore, a primitive sample analysis was performed to identify applications using multiple platforms.(wt)

Based on the analysis, most popular applications specify one or more platforms in their specifications. Approximately 33.33% of the 100M+ applications did not specify the platform, and the percentage decreased for the remaining groups (50M +, 5M +, and 1M +) to approximately 30.70%. Cross-platform ability is available in more than half of the highest downloaded applications group at 54.55%.

Regarding the whole sample, more than half of the applications in the sample did not specify any platform for their applications, 58.82%. Applications that choose the phone as their platform regardless of the tablet are 132 applications in 32. 35% while applications that choose the tablet as their platform regardless of the phone are 167 applications in 40. 93%. Most of the applications in the sample do not support the cross-platform capability of the applications, where only 133 applications support this ability at 32.60%.

A summary of the applications across the different metadata modalities is shown in Table 5. Based on the analysis of 408 APKs, different metrics are presented in Table 5. The average column represents the mean size per app for the given feature modality. For instance, for the CFG modality, the table shows the average number of .dot files per application, where .dot files were extracted from the dataset using Jadex analysis tool. These graphs are disconnected graphs that have disconnected components. For functions and strings, the average refers to the average number

of functions and strings per APK, respectively. For hexdump files, the size and representation can vary significantly depending on the content of the APK files and how the hexdump is generated and formatted. The raw hexdump is hexadecimal values capturing the size of the program, ranging from 105,475 bytes to 1,541,415,915 bytes, and the eventual dimensionality depends on the representation (hexadecimal, n-grams, etc.). API Calls average illustrates the average frequency of external interactions. Lastly, the Metadata has 5 features: malware detection, safety landscape, audience, platforms, and app domains. Each feature contains multiple values.

3.4.8 Case Study: AR/VR Permission Analysis

In the following, we highlight one case study of our dataset centered around app permission analysis to understand affinities and potential anomalies. In our study, We utilized AAPT2 to read the APK files, extract the AndroidManifest.xml, parse the permissions, and save them into a CSV file. This tool enabled us to systematically analyze and catalog the permissions requested by each AR/VR application in our dataset. The CSV file will contain the APK names and the associated extracted permissions.

To gain insights into the frequency and prevalence of specific permissions across different age groups, a feature in our dataset, we performed a detailed analysis as follows:

- **Loading Permissions**: We extracted permissions for each APK and stored them in a structured format (CSV). This allowed for efficient querying and analysis.
- **Frequency Analysis**: We computed the frequency of each permission across the entire dataset to identify the most commonly requested permissions essential for AR/VR applications operation.
- **Prevalence by Age Group**: We categorized the applications into different age groups (Everyone, Everyone +10, Teen, Mature) based on the age-based rating shown in Table 3 and included the whole sample's permissions frequency for comparison. We determined the top 5 most prevalent permissions for all five aspects. This analysis highlights how permissions requirements vary with the intended audience and compares them with the permissions for the dataset.
- **Cleaning and Standardizing Permissions**: To ensure consistency and clarity, we cleaned and standardized the permissions, removing unnecessary text and focusing on the core permission entity. However, during our analysis, we encountered permissions that did not follow the standard Android permission format, such as *com.resolutiongames.codenamelazarus* and *com.arfps.android* (custom 1 and custom 2 in Table 6). These unique or custom permissions suggest specialized functionalities within those specific applications.

Customized permissions can potentially pose security threats due to their lack of transparency, potential for privilege escalation, and reliance on secure implementation. To mitigate these risks, developers could minimize custom permissions, provide clear documentation, and ensure secure implementation. For users, such an analysis could inform them by highlighting irregularities and deviations in permission requests from the larger app populations.

3.5 Limitation

Various limitations were encountered when collecting and analyzing the dataset. One of these limitations is the limited size of the sample, which limits the generalization to all AR/VR applications. This study's sample size limitation is due to several restrictions, such as the platform, geographical regions, and emulator compatibility.

Some applications undergo frequent updates, so the data presented on them may change over time. Although each application was acquired with its version number, this does not guarantee that the updated version will be available after it has been updated or removed. More sophisticated analysis tools may be used to examine applications that provide additional data or present the results differently.

This study analyzed free applications and excluded paid ones. As the dataset from this study is provided, retroactively running the updated analysis tools should not require much effort. Moreover, AR/VR applications for mobile devices have become increasingly popular. As noted, more applications have appeared in the Google Play store, and future research may be able to utilize a larger sample size.

It is recommended that AR/VR applications across platforms be examined and analyzed to obtain as much raw data as possible and for generalization. Researchers should continue investigating the challenges encountered when creating and maintaining AR/VR applications, as some AR/VR apps have not been updated in more than three years and remain available in the Google Play store. Moreover, it is important to emphasize the direction and trend that AR/VR applications are following to identify which aspects require improvement and which are currently gaining traction. For example, the dataset indicates a significant trend in education and entertainment. A decrease in games was also observed compared to education and entertainment during the same period.

3.6 Summary

In this study, we addressed the significant gap in the availability of comprehensive AR/VR application datasets by developing a robust dataset from 408 diverse applications sourced from the Google Play Store. Our work standardized multiple data modalities, including control flow graphs, strings, functions, permissions, API calls, hexdump, and metadata, facilitating a broad spectrum of security analyses. The results demonstrated the dataset's utility in enhancing AR/VR security

evaluation, providing researchers with a valuable tool for further studies.

Despite the extensive data collection, the study acknowledges limitations such as the representation of applications exclusively on one platform. This may affect the generalizability of the findings across other mobile ecosystems. Future research could expand this dataset by including applications from various platforms and integrating additional data modalities. This would cover more comprehensive aspects of security and performance evaluation.

There is also an opportunity to apply machine learning techniques to predict potential security vulnerabilities based on the dataset. This could profoundly impact the development of more secure AR/VR applications. While no further work is planned for this particular dataset, the research community is encouraged to leverage this foundation for more detailed and expansive studies in the AR/VR domain.

4 x-scope: LLM-Based Analysis of AR/VR Android Applications Privacy Policies

4.1 Summary of Completed Work

In our research, we employed BERT, a state-of-the-art NLP model, to analyze AR/VR privacy policies. By comparing these policies with those of free and premium content websites, we identified key differences in the clarity and thoroughness of the information provided. Our findings indicate that AR/VR applications generally offer clearer privacy policies than free content websites, but they still lag behind premium sites. This analysis has provided valuable insights into how privacy policies can be structured to enhance user understanding and trust. This has contributed to the ongoing discussion about privacy practices within the AR/VR industry.

4.2 Introduction

Privacy policies are critical documents that inform users about how their personal data is collected, used, and shared by applications [6, 7, 100]. Despite their importance, privacy policies are often long, complex, and difficult for users to understand [43, 50, 100]. This complexity can lead to privacy policy violations where an application's actual data practices do not align with its stated policies [74]. Various tools and methods have been proposed to automatically analyze privacy policies and detect potential violations [15, 26]. For example, PVDetector [74] maps descriptive phrases in privacy policies to privacy-related API invocations in the corresponding app code. These tools help in identifying misalignments that may have legal repercussions [8].

These policies are especially pertinent in the context of AR/VR applications, which often collect a wide range of personal and sensitive data, including biometric information, user interactions, and environmental details [53, 60]. As AR/VR technologies continue to grow in popularity and usage, ensuring that privacy policies are clear, comprehensive, and transparent is essential to maintaining user trust [41] and compliance with regulatory standards [42]. However, privacy policies are often complex and confusing, making it difficult for users to fully understand the implications of their data being collected and used [54]. Additionally, privacy policies can be excessively long and overwhelming, further discouraging users from reading and comprehending them [53, 71].

AR/VR applications have an exceptional position in the technology landscape due to their immersive nature and interaction level [53]. These applications are not only used for entertainment and gaming but also in education, healthcare, retail, and industrial training. These applications collect highly sensitive data, making it imperative for developers to implement robust privacy practices. In addition, developers must communicate these practices effectively through well-crafted privacy policies.

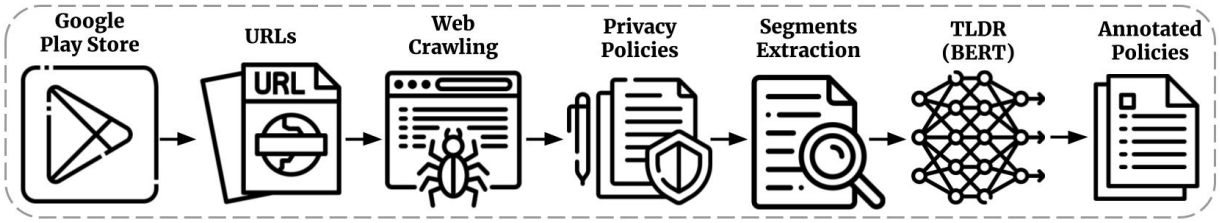

Figure 4: Privacy Policy Pipeline

Despite the critical importance of privacy policies, limited research has focused on analyzing AR/VR privacy practices [53]. This study aims to fill this gap by leveraging advanced text classification models like BERT. It assesses the transparency and thoroughness of privacy policies in AR/VR applications. By comparing these policies with those of free and premium websites, a comprehensive understanding of privacy practices in the AR/VR industry is sought. This comparison also is designed to identify areas for improvement.

4.3 Privacy Policy Analysis Pipeline

Our methodology systematically collects and processes privacy policies from AR/VR applications, employing advanced NLP techniques. BERT was tuned using an annotated dataset containing segments of privacy policies, which had been meticulously curated. This fine-tuning ensured that the analysis maintained alignment with prior research and achieved a semantically robust mapping of policies to high-level attributes.

This assessment within AR/VR applications was conducted by categorizing and evaluating positive segments in which specific policy elements can be identified and highlighting segments and key terms. Furthermore, this approach enabled us to assess the richness and expressiveness of these privacy policies compared to those of other domains, such as website privacy policies. *x-droid* analysis provides critical insights into the nuances of privacy policy articulation in AR/VR applications. This contributes to the broader understanding of privacy practices in emerging technological contexts.

4.3.1 Dataset Scraping & Transformation

Each app's privacy policy was extracted using advanced web scraping techniques. Initially, 408 URLs for 408 AR/VR apps from [14] were collected and subsequently filtered to ensure relevance and accessibility. The automated approach minimized manual effort and enhanced accuracy, allowing for the efficient compilation of a comprehensive list of privacy policy URLs.

The next phase involved retrieving the HTML content of the privacy policies using the provided URLs. This process addressed various edge cases, such as non-English content and unavailable pages. Privacy policies were sometimes only available as images, lacked accessible links or documents, or were not in English. Additionally, some applications were no longer available on the

Table 7: Availability of privacy policies in *x-droid*.

Type	Included?	Applications
Image	✖	1
No link or document	✖	16
Not in English	✖	47
Apps not available	✖	42
Text	✔	22
HTML	✔	280

Google Play Store. These challenges required robust error handling and verification mechanisms to ensure the integrity and completeness of the collected data.

After addressing these edge cases, 302 usable privacy policies were obtained. The process involved reading URLs from structured data files and systematically retrieving HTML content for each URL. This automated approach ensured a systematic collection and storage of all accessible privacy policies for subsequent analysis. The statistics of the final privacy policies across types are presented in Table 7, highlighting the inclusion of 302 text or HTML format policies.

For data pre-processing, advanced parsing techniques was employed to extract meaningful content from HTML files. Using robust tools, each HTML file was parsed individually to extract relevant text elements. These elements were then standardized and stored in a format suitable for further analysis. This pre-processing step was crucial in preparing the data for in-depth examination, allowing us to focus on the substantive content of the privacy policies. The pre-processing steps included the removal of extraneous elements to ensure clean and usable text data. This comprehensive extraction and transformation process is visualized in Figure 4.

4.3.2 Dataset Processing

The extracted text from privacy policies was tokenized into individual words, with non-alphabetic tokens filtered out. Stopwords were removed to highlight meaningful words. Key privacy-related terms such as information, personal, data, policy, service, and privacy were quantified. These frequency offered a foundational understanding of the prevalence of critical privacy terms across various policies. To ensure dataset quality, a filtering mechanism was implemented based on the presence of essential privacy-related keywords. Text files lacking the term ‘privacy’ were deemed irrelevant and excluded. Additionally, files with fewer than two instances of key terms such as ‘information,’ ‘personal,’ ‘data,’ and ‘policy’ were removed. This filtering process involved leveraging previously saved word count data to identify and eliminate non-relevant files, refining the dataset to include only substantive policies.

Further refinement was achieved by excluding short, non-informative paragraphs (*i.e.,* fewer than five words). This process involved systematically iterating through each text file, identifying and removing short segments to ensure the retention of more substantive content. The cleaned

text was subsequently saved, focusing on enhancing data quality for robust analysis. To facilitate comprehensive analysis, text files from each application were consolidated into a single document. This step was necessary to efficiently manage large volumes of text data, providing a holistic view of each privacy policy and enabling more effective content processing.

All paragraphs within the text files were annotated to enhance the analysis further. This involved converting paragraphs to lowercase and storing them in a structured format, ensuring efficient access and processing in subsequent research stages. The annotated paragraphs were crucial for organizing and preparing the data for advanced analyses such as topic modeling and sentiment analysis. The total number of paragraphs and words across all documents was aggregated and analyzed, with average metrics per document calculated. This was done to understand the general structure and verbosity of privacy policies. These aggregated statistics were preserved for detailed dataset overview and analysis.

For comparative analysis, different groups (e.g., categories or types of applications) were examined. Furthermore, the number of paragraphs and words for each group was calculated, leveraging the annotated data to count and average these metrics. The results were saved in a structured format, facilitating easy access and comparison across groups. This step enabled us to identify trends and differences in privacy policy disclosures among various application types, enhancing the understanding of privacy practices in different contexts.

4.3.3 Privacy Policy Categories

The methodology developed by [9, 10] was applied to categorize privacy policy segments into nine high-level categories: First Party Use, Third Party Sharing, User Choice, User Access, Data Retention, Data Security, Policy Change, Do Not Track, and Specific Audiences [8]. This framework was originally used to analyze the privacy policies of free content and premium websites. It is highly relevant to the study of AR/VR apps' privacy policies. By utilizing these categories, the primary objective is to uncover the specific privacy practices and commitments of AR/VR apps, providing a detailed comparison with the findings from [10].

First Party Use includes segments that describe how the app developer uses the collected data [85] internally. This category is essential for understanding the primary purposes for which user data is acquired and processed by the AR/VR app itself. Third Party Sharing comprises segments outlining how the app shares user data with external entities [68]. This category helps to identify potential privacy risks associated with data dissemination beyond the primary app developer. User Choice encompasses segments that describe users' options regarding their data [54]. This includes opt-in and opt-out mechanisms, allowing users to control how their data is collected and used.

User Access includes segments that explain how users can access and manage their own data [68]. This category is critical for assessing the privacy policy's transparency and user empowerment aspects. Data retention includes segments that describe how long user data is retained

by the app [56]. This category helps understand the data lifecycle and app data retention policies. Data Security encompasses segments that describe the app's measures to protect user data [85]. This category is crucial for evaluating the robustness of the app's data protection practices.

Policy change includes segments that explain how users will be informed about privacy policy changes [68]. This category is relevant for understanding the app's commitment to transparency and user notification. Do Not Track features segments that describe the app's stance on tracking signals and mechanisms to respect user preferences regarding tracking [54]. This category helps assess the app's adherence to user privacy preferences. Specific Audiences includes segments that describe data handling practices targeted at specific user groups, such as children or other vulnerable populations [85]. This category is significant for evaluating how the app addresses privacy needs of different demographics.

4.3.4 BERT Analysis

BERT (Bidirectional Encoder Representations from Transformers) is a language representation model developed by Google [34]. It excels in various NLP tasks due to its ability to consider context from both directions in a sentence [7, 36]. BERT's bidirectional nature allows it to capture nuanced meanings and relationships within text, making it highly effective for analyzing complex privacy policies. BERT was selected for the analysis mainly due to its superior performance in understanding and classifying text, particularly in tasks requiring deep contextual comprehension. This is essential for accurately interpreting privacy policy documents.

The BERT model was fine-tuned using data from [10]. The training process was conducted using the Ktrain library in Google Colab, facilitating efficient training and evaluation of the model [23]. The necessary data, including pre-processed paragraphs, category labels, and document lengths, was used to train the model. *x-droid* training input design mirrored that utilized by [10], ensuring methodological consistency and enabling a meaningful comparison between *x-droid* results and those of [10]. The input values include: 1. *Category Labels:* Labels for each paragraph, indicating each segment's high-level category, as defined in the OPP-115 dataset used by [10]. 2. *Training Paragraphs:* Text segments from the OPP-115 dataset include annotated segments from privacy policies labeled according to the nine categories identified by [10]. 3. *Length:* The number of text segments in each document used to maintain the context of the documents during training.

The training process involved fine-tuning hyperparameters, including learning rate, batch size, and epochs, to optimize model performance [34]. Data augmentation techniques were employed to increase robustness and generalizability. Additional training data was generated by modifying existing data, and cross-validation was utilized by validating the model with different subsets of the training data [34]. A custom function was used to split the dataset into training and testing sets [36] to ensure a balanced representation of categories. This approach enabled the BERT model

to accurately classify privacy policy segments into the predefined categories set by [10], effectively reflecting various privacy practices.

The training includes the following steps: 1. Loading Data: Data, including paragraphs, category labels, and document lengths, was loaded into the training environment. 2. Data Splitting: The dataset was divided into training and testing sets while maintaining a balanced distribution of categories to ensure representativeness. 3. pre-processing: The training and testing data underwent pre-processing using BERT-specific tokenization and formatting techniques. 4. Model Training: The BERT model was trained on the training data, with hyperparameters fine-tuned to optimize performance. 5. Model Evaluation: The trained model was evaluated on the testing data to assess its accuracy and generalizability.

The BERT model was employed to classify and analyze segments. The model's output included classification labels for each segment, enabling the identification and categorization of various privacy practices. This automated classification facilitated a detailed and systematic analysis of the privacy policies, highlighting key practices and commitments.

Positive segments, as defined in *x-droid* study, are clear statements explaining data handling practices. These segments were identified using BERT and are crucial for understanding the transparency and comprehensiveness of the privacy policies. By focusing on positive segments, the effectiveness of the policies in communicating sensitive privacy information to users was assessed. The identification process involved calculating the percentage of positive segments within each category. For each category, the number of paragraphs containing at least one positive statement was determined and compared to the total number of paragraphs, providing a clear metric of transparency and detail in the privacy policies. This analysis quantifies the extent to which privacy policies explicitly communicate their data-handling practices to users.

Highlighted segments and specific words within the privacy policies were analyzed to identify key privacy practices and commitments. Highlighted segments refer to specific parts of the text emphasized either by the policy itself or through the analysis as particularly relevant. Analyzing highlighted segments involved calculating the percentage of significant segments within each category. The number of highlighted segments was calculated and compared to the total number of paragraphs for each category. This provided a metric for the emphasis placed on key privacy practices. This analysis helped pinpoint critical data collection, sharing, retention, and security information. Highlighted segments facilitated the identification of the most significant aspects of each privacy policy, ensuring essential information was not overlooked.

Highlighted words are key terms within the privacy policies critical for understanding the policies' content regarding data collection, sharing, retention, and security. Analyzing highlighted words involved calculating the percentage of significant words within each category. For each category, the number of highlighted words was summed and compared to the total number of words in that category. This analysis quantifies the emphasis placed on key terms within privacy policies.

Table 8: The statistics of *x-droid*'s dataset.

Metric	Count
Total Policies	240
Total Paragraphs	25,135
Avg. Paragraphs	104.73
Total Words	930,225
Avg. Words	3,875.94

These aspects were chosen based on the framework used by [10], as they represent fundamental components of privacy practices that must be communicated effectively to users. This analysis helped pinpoint critical data collection, sharing, retention, and security information. Highlighted words facilitated the identification of the most relevant aspects of each privacy policy, ensuring essential information was not overlooked. This step was essential for evaluating privacy policies' effectiveness in conveying crucial information to users. It also identified any gaps or areas for improvement.

4.4 Results & Discussion

In this section, the results of the analysis of AR/VR application privacy policies are presented, leveraging BERT for classification and evaluation. Various aspects are examined, including word and paragraph counts, BERT training accuracy, positive segments, highlighted segments, and highlighted words. By comparing these findings with the results from [10], insights into the transparency and comprehensiveness of privacy policies in the AR/VR domain are provided. This analysis highlights the current state of privacy practices in AR/VR applications and identifies areas for improvement.

4.4.1 Words and Paragraphs Count

Overall. The analysis began with a comprehensive examination of the total word and paragraph counts in AR/VR application privacy policies. A total of 240 policies were processed, resulting in a dataset containing 25,135 paragraphs and 930,225 words. This yields an average of 104.73 paragraphs and 3,875.94 words per policy. These metrics indicate that privacy policies for AR/VR applications tend to be relatively detailed. This is consistent with the need for thorough explanations of data practices in technology-intensive domains.

Comparatively, the average length of AR/VR privacy policies, in terms of both paragraphs and words, aligns more closely with the detailed policies of premium websites analyzed by [10], which often feature comprehensive privacy disclosures. This suggests that AR/VR applications, similar to premium websites, prioritize detailed privacy statements to address unique data practices.

Groups. To gain a deeper understanding of the variations in privacy policy content across different

types of AR/VR applications, the policies were categorized into specific groups based on their primary functions. These groups included education, games, entertainment, simulation, tools, video players, business, art & photo, casual, books & news, social & communication, lifestyle, productivity, sports, health, travel & maps, and shopping.

x-droid findings revealed significant variations among these groups. For instance, the Entertainment group exhibited the highest average number of paragraphs per policy at 260.6, indicating particularly detailed and extensive privacy policies. In contrast, the Travel & Maps group had the lowest average at 25 paragraphs, suggesting shorter privacy policies. Similarly, in terms of word counts, the Entertainment group had the highest total word count (297,159), reflecting its high average paragraph count, while Travel & Maps had the lowest word count (393), aligning with its fewer paragraphs.

These disparities highlight how different AR/VR application categories prioritize privacy disclosures. Applications in categories such as entertainment and games, which may involve more complex data interactions, tend to provide more detailed privacy policies. Conversely, categories like travel and maps might have simpler data practices, resulting in shorter policies.

Table 9: Group-based statistics of *x-droid*'s dataset.

Group	Policies	Para.	μ Para.	Words	μ Words
Education	48	3,172	66.08	122,376	2,549.50
Games	30	7,873	262.43	221,499	7,383.31
Entertainment	30	7,818	260.60	297,159	9,905.30
Simulation	28	1,441	51.46	56,676	2,024.86
Tools	21	1,755	83.57	68,557	3,264.62
Video Players	18	484	26.89	21,960	1,220.00
Business	18	163	9.06	7,406	411.44
Art & Photo	18	140	7.78	9,716	539.78
Casual	18	450	25.00	15,678	871.00
Books & News	18	488	27.11	18,536	1,029.78
Social & Comm.	16	359	22.44	14,527	908.94
Lifestyle	15	153	10.20	6,915	461.00
Productivity	15	422	28.13	15,774	1,051.60
Sports	10	175	17.50	7,455	745.50
Health	10	141	14.10	6,750	675.00
Travel & Maps	1	25	25.00	393	393.00
Shopping	5	76	15.20	3,417	683.40

4.4.2 BERT Training

Evaluation Metrics. The BERT model was evaluated using multiple metrics to ensure a comprehensive assessment of its performance. Accuracy was determined as the proportion of correctly

classified segments out of the total number of segments. Precision and recall were calculated for each category. Precision indicates the proportion of true positive predictions among all positive predictions, and recall reflects the proportion of true positive predictions among all actual positives. The F1-score, which is the harmonic mean of precision and recall, was utilized to provide a balanced measure of the model's performance. The overall performance of the model was assessed by averaging these metrics across all categories, offering a holistic view of its classification capabilities.

Table 10: Comparison of the accuracy with different works.

Category	***x-droid***	**TLDR [10]**	**Wilson [85]**	**Harkous [43]**	**Liu [54]**
1st Party	0.93	0.94	0.75	0.79	0.81
3rd Party	0.93	0.89	0.70	0.79	0.79
User Choice	0.96	0.85	0.61	0.74	0.70
User Access	0.98	0.91	0.61	0.80	0.82
Data Retention	0.99	0.87	0.16	0.71	0.43
Data Security	0.98	0.88	0.67	0.85	0.80
Policy Change	0.99	0.95	0.75	0.88	0.85
Do Not Track	1.00	1.00	1.00	0.95	1.00
Audiences	0.98	0.94	0.70	0.95	0.85
Overall	0.97	0.91	0.66	0.83	0.78

Training Process and Results. The BERT model was initialized by downloading the pretrained BERT model (*uncased_L-12_H-768_A-12.zip*) and extracting it for use. The training data was preprocessed with a maximum sequence length of 512 tokens, and the model was fine-tuned using the onecycle policy with a maximum learning rate of 2e-05.

During training, the model's performance improved significantly over the epochs across all categories. For instance, in the 1st Party Use category, the model's accuracy increased from 80.5% in the first epoch to 99.68% in the final epoch. Similarly, the Do Not Track category demonstrated remarkable performance with an accuracy of 100% achieved in several epochs. The detailed training results for all categories are summarized below:

- **1st Party Use**: Initial accuracy of 80.5% in epoch 1, improving to 99.68% in epoch 10.
- **3rd Party Sharing**: Initial accuracy of 78.17% in epoch 1, improving to 99.84% in epoch 10.
- **User Choice**: Initial accuracy of 91.53% in epoch 1, improving to 99.87% in epoch 10.
- **User Access**: Initial accuracy of 96.79% in epoch 1, improving to 99.84% in epoch 10.
- **Data Retention**: Initial accuracy of 95.86% in epoch 1, improving to 99.91% in epoch 10.
- **Data Security**: Initial accuracy of 93.65% in epoch 1, improving to 99.90% in epoch 10.

- **Policy Change**: Initial accuracy of 97.16% in epoch 1, improving to 99.87% in epoch 10.
- **Do Not Track**: Initial accuracy of 99.24% in epoch 1, achieving 100% accuracy in several epochs.
- **Specific Audiences**: Initial accuracy of 95.26% in epoch 1, improving to 99.87% in epoch 10.

Reasons for Higher Results. Despite using the same input data and similar training code as [10], *x-droid* BERT model achieved higher performance metrics. Several factors could contribute to this difference, including the hyperparameter tuning, the data augmentation and preprocessing, model initialization, training environment, and regularization technique.

4.4.3 Positive Segments

In this section, positive segments identified in AR/VR privacy policies are analyzed. Positive segments are those paragraphs that clearly articulate privacy practices and policies in a positive light, providing transparency and reassurance to users. The analysis results are compared with those from [10], examining both overall trends and specific group differences.

Overall Comparison. Table 11 presents the overall percentage of positive segments across all categories. The results indicate that AR/VR applications have a higher percentage of positive segments than free content but a lower percentage than premium websites. This reflects a greater emphasis on transparency and user trust in premium websites compared to AR/VR applications.

Table 11: Comparison of the distribution of the positive segments across various categories. Δ_F captures the difference between the distribution in *x-droid* and the other group.

Category	***x-droid***	**Free**	Δ_F	**Premium**	Δ_P
1st Party Use	97.08	86.90	+10.18	95.73	+1.35
3rd Party Sharing	92.50	84.52	+7.98	89.69	+2.81
User Choice	57.92	52.38	+5.54	79.27	-21.35
User Access	39.58	50.00	-10.42	65.81	-26.23
Data Retention	42.08	30.95	+11.13	57.26	-15.18
Data Security	75.00	67.86	+7.14	75.00	0.00
Policy Change	83.33	71.43	+11.90	72.22	+11.11
Do Not Track	14.17	12.70	+1.47	21.58	-7.41
Specific Audiences	77.50	67.86	+9.64	74.15	+3.35
Average	64.35	58.29	+6.06	70.08	-5.73

The comparison shows that AR/VR applications have a higher percentage of positive segments than free content but a lower percentage than premium websites. This suggests that while AR/VR applications are more transparent and provide better privacy assurances than free content, they

still lag behind premium websites regarding overall positive segments. Categories like 1st Party Use, 3rd Party Sharing, Data Retention, and Policy Change exhibit significant positive differences between AR/VR apps and both free and premium websites.

The average value presented in Table 11 is the mean value of all categories. It is calculated by summing the percentages of all categories and dividing by the total number of categories. This average provides a general overview of how positive segments are distributed across different privacy policy categories.

Groups Comparison. In addition to the overall comparison, the positive segments for different groups of AR/VR applications were also analyzed. Table 12 shows the percentage of positive segments for each group.

Table 12: Comparison of the mean distribution value (μ) of the positive segments in *x-droid* groups and other groups.

Category	μ ***x-droid***	μ **Free**	μ **Premium**
1st Party Use	98.28	89.69	94.64
3rd Party Sharing	85.05	86.69	89.00
User Choice	64.86	59.06	79.71
User Access	38.82	44.33	67.20
Data Retention	38.73	34.92	59.61
Data Security	69.58	67.97	75.78
Policy Change	75.19	72.41	71.75
Do Not Track	8.82	12.25	22.87
Specific Audiences	63.72	68.97	75.68

The group comparison reveals that AR/VR applications generally perform better or on par with free websites in terms of positive segments. However, there are some categories, such as Do Not Track and Specific Audiences, where AR/VR apps have lower percentages than other domains.

Observations and Explanations. The high percentage of positive segments in AR/VR applications can be attributed to several factors: 1. Increased Focus on Transparency: AR/VR applications often handle more sensitive data and have a higher level of user interaction, necessitating an increased focus on transparency to build user trust. 2. Regulatory Compliance: Stricter privacy regulations and guidelines for AR/VR technologies may encourage developers to provide more comprehensive and clear privacy policies. 3. User Expectations: Users of AR/VR applications may have higher expectations regarding privacy, prompting developers to be more transparent about their data practices.

Overall, *x-droid* analysis indicates that AR/VR applications are making significant strides in privacy transparency. This is evidenced by the higher percentage of positive segments than free content and the targeted emphasis on critical privacy practices. However, they still have room to improve compared to premium websites.

4.4.4 Highlighted Segments

This section focuses on analyzing highlighted segments within privacy policies for AR/VR applications. Highlighted segments refer to those parts of the privacy policy that explicitly emphasize key privacy practices. This is often through bold text, headings, or other visual markers. The results are compared with those from [10] to evaluate the prominence and clarity of privacy practices in AR/VR applications.

Overall Comparison. Table 13 presents the overall percentage of highlighted segments across all categories. The results suggest that AR/VR applications have a varied distribution of highlighted segments compared to free and premium websites.

Table 13: Comparison of the distribution of the highlighted segments in *x-droid* and other groups.

Category	***x-droid***	**Free**	Δ_F	**Premium**	Δ_P
1st Party Use	35.51	25.76	+9.75	32.91	+2.60
3rd Party Sharing	19.06	16.00	+3.06	15.77	+3.29
User Choice	4.60	5.70	-1.10	6.12	-1.52
User Access	1.45	3.23	-1.78	3.14	-1.69
Data Retention	2.12	2.43	-0.31	1.89	+0.23
Data Security	3.54	3.39	+0.15	2.62	+0.92
Policy Change	2.57	2.22	+0.35	2.65	-0.08
Do Not Track	0.21	0.45	-0.24	0.31	-0.10
Specific Audiences	4.08	7.34	-3.26	8.37	-4.29
Overall	62.29	58.96	+3.33	64.33	-2.04

The overall comparison reveals that AR/VR applications generally perform similarly to or slightly better than free websites in terms of highlighted segments. However, they still lag behind premium websites in some categories. Notable categories with higher percentages include 1st Party Use and 3rd Party Sharing, indicating that these areas are more prominently emphasized in AR/VR applications.

Groups Comparison. In addition to the overall comparison, the highlighted segments for different groups of AR/VR applications were also examined. Table 14 shows the percentage of highlighted segments for each group.

The group comparison indicates that AR/VR applications have varying levels of highlighted segments across different categories. While categories like 1st Party Use and 3rd Party Sharing show higher emphasis, User Choice and User Access reveal lower percentages than free and premium websites.

Observations and Explanations. The distribution of highlighted segments in AR/VR applications suggests a targeted approach to emphasizing specific privacy practices. Possible reasons include: 1. Targeted Emphasis: Developers might prioritize highlighting critical privacy practices relevant to AR/VR users. 2. Regulatory Focus: Emphasis on certain categories could be driven

Table 14: Comparison of the mean distribution value (μ) of the highlighted segments in *x-droid* groups, free websites groups, and premium websites groups privacy policies.

Category	μ *x-droid*	μ Free	μ Premium
1st Party Use	98.28	89.69	94.64
3rd Party Sharing	85.05	86.69	89.00
User Choice	64.86	59.06	79.71
User Access	38.82	44.33	67.20
Data Retention	38.73	34.92	59.61
Data Security	69.58	67.97	75.78
Policy Change	75.19	72.41	71.75
Do Not Track	8.82	12.25	22.87
Specific Audiences	63.72	68.97	75.68

by regulatory requirements specific to AR/VR technologies. 3. User Experience: Enhanced user experience in AR/VR applications may lead developers to emphasize key privacy segments for better comprehension.

Overall, AR/VR applications exhibit a strategic approach to privacy segments, reflecting a comprehensive understanding of user needs and regulatory demands.

4.4.5 Highlighted Words

Introduction. In this section, highlighted words in AR/VR privacy policies are evaluated. Highlighted words are those that are frequently used and emphasized in the context of privacy, such as 'data,' 'personal,' and 'information.' The frequency and emphasis of these words are compared with the results from [10] to assess privacy communications focus areas.

Overall Comparison. Table 15 presents the overall percentage of highlighted words across all categories. The results highlight the emphasis on certain key terms in AR/VR applications compared to free and premium websites.

The comparison shows that AR/VR applications generally have a higher percentage of highlighted words in categories like 3rd Party Sharing, Data Security, and Policy Change than free and premium websites. This reflects a focused effort to emphasize key privacy-related terms in these areas.

Groups Comparison. In addition to the overall comparison, the highlighted words for different groups of AR/VR applications were also evaluated. Table 16 shows the percentage of highlighted words for each group.

The group comparison highlights that AR/VR applications emphasize certain privacy-related terms more consistently across different groups. Categories like 3rd Party Sharing and Data Security show higher percentages of highlighted words, indicating a strong emphasis on these aspects.

Observations and Explanations. The focus on highlighted words in AR/VR applications can

Table 15: Comparison of the distribution of the highlighted words in *x-droid* and other groups. All numbers are %.

Category	*x-droid*	Free	Δ_F	Premium	Δ_P
1st Party Use	36.05	40.45	-4.40	31.41	+4.64
3rd Party Sharing	28.46	19.15	+9.31	21.04	+7.42
User Choice	6.04	6.29	-0.25	6.42	-0.38
User Access	1.86	3.56	-1.70	3.96	-2.10
Data Retention	1.97	2.39	-0.42	3.40	-1.43
Data Security	5.43	2.72	+2.71	4.29	+1.14
Policy Change	4.16	3.07	+1.09	2.84	+1.32
Do Not Track	0.24	0.27	-0.03	0.49	-0.25
Specific Audiences	5.47	8.44	-2.97	10.71	-5.24
Overall	75.58	69.37	+6.21	71.01	+4.57

Table 16: Comparison of the mean distribution value (μ) of the highlighted words in *x-droid* groups, free websites groups, and premium websites groups privacy policies.

Category	μ *x-droid*	μ Free	μ Premium
1st Party Use	98.28	89.69	94.64
3rd Party Sharing	85.05	86.69	89.00
User Choice	64.86	59.06	79.71
User Access	38.82	44.33	67.20
Data Retention	38.73	34.92	59.61
Data Security	69.58	67.97	75.78
Policy Change	75.19	72.41	71.75
Do Not Track	8.82	12.25	22.87
Specific Audiences	63.72	68.97	75.68

be attributed to several factors: 1. Key Term Emphasis: Highlighting specific terms helps clearly communicate critical privacy practices to users. 2. Regulatory Requirements: Emphasizing certain terms might be driven by regulatory guidelines that mandate clear communication of privacy practices. 3. User Trust: Using highlighted words to emphasize key privacy aspects can enhance user trust and confidence in the application.

Overall, AR/VR applications demonstrate a deliberate effort to highlight key privacy terms, enhancing their privacy policies' clarity and effectiveness.

4.5 Key Findings, Limitations, and Future Work

4.5.1 Key Findings

The analysis revealed that AR/VR applications generally have a higher percentage of positive segments than free content but lower than premium websites. This suggests that AR/VR applications place greater emphasis on transparency and user trust than free content, but there is still room for improvement compared to premium websites. Categories such as 1st Party Use, 3rd Party

Sharing, and Policy Change demonstrated substantial positive differences, highlighting AR/VR applications' proactive approach to user privacy concerns.

The analysis of highlighted segments showed that AR/VR applications strategically emphasize critical privacy practices. Categories such as 1st Party Use and 3rd Party Sharing are more prominently highlighted in AR/VR applications than in free websites, though less emphasized than premium websites. This targeted emphasis likely enhances the visibility and clarity of privacy information.

Furthermore, AR/VR applications consistently use highlighted words to emphasize key privacy terms like 'data,' 'personal,' and 'information.' This deliberate focus on highlighting crucial terms aids in clearly communicating privacy practices, contributing to greater user trust and regulatory compliance. Categories like 3rd Party Sharing and Data Security show higher percentages of highlighted words, reflecting a strong emphasis on these aspects.

Overall, the analysis underscores the significant strides AR/VR applications are making in enhancing privacy transparency and user trust. The use of BERT for text classification has proven highly effective, yielding accurate and insightful results. AR/VR applications demonstrate a strong commitment to clear and comprehensive privacy practices. This is evidenced by the higher percentage of positive segments, targeted highlighted segments, and emphasized key terms.

These findings highlight the evolving landscape of privacy practices in AR/VR applications, offering valuable insights for developers, regulators, and users. *x-droid* study provides a robust framework for further research and development in privacy policy analysis, paving the way for more transparent and user-friendly privacy practices in emerging technologies.

4.5.2 Limitations and Future Work

Despite the promising results obtained from *x-droid* analysis, several limitations must be addressed in future research. One primary limitation is the dataset scope. *x-droid* study focused on a specific set of AR/VR applications. Expanding the dataset to include a more diverse application, including different regions, would provide a more comprehensive understanding of privacy practices across the AR/VR industry.

Another limitation lies in automated tools for text classification and analysis. While BERT has proven effective in *x-droid* study, the model's performance could be further improved by incorporating more advanced techniques and fine-tuning with a larger and more diverse dataset [34]. Additionally, the reliance on predefined privacy categories might not capture the full spectrum of privacy concerns users may have [23].

Future work should also consider the dynamic nature of privacy policies. Privacy practices and regulations are continually evolving, and it is crucial to keep the analysis up-to-date with the latest changes [41]. Developing automated systems that can regularly update and re-evaluate privacy policies would ensure that the findings remain relevant and accurate over time.

Finally, user studies could be conducted to evaluate how well the highlighted segments and words improve user understanding. A user's perception and behavior regarding privacy policies can provide valuable feedback for improving AR/VR privacy information presentation.

4.6 Summary and Work to be Completed

This research examined the clarity and thoroughness of privacy policies in AR/VR applications by employing BERT, a state-of-the-art NLP model. The findings indicate that AR/VR applications generally offer more detailed privacy policies than free content websites but do not meet premium sites' standards. This analysis highlights significant gaps in privacy policy practices within the AR/VR industry, emphasizing the necessity for enhanced privacy disclosures.

This study focuses exclusively on Android platforms. This restricts the generalizability of the results across different operating systems and application environments. Future research should broaden the scope to include AR/VR applications on other platforms, such as iOS and Windows. It should also consider AR/VR-specific privacy policies.

Work to be Completed. I will expand the scope of my research by collecting and analyzing privacy policies from a broader array of AR/VR platforms. These platforms include iOS, Windows, and AR/VR-specific websites. I plan to utilize the BERT model to evaluate this expanded dataset, allowing for a direct comparison with initial findings from Android-based AR/VR applications. My goal is to deliver a comprehensive overview of privacy practices across these varied platforms. I also want to enhance privacy management understanding within diverse AR/VR environments.

5 x-API: Unified API Call-based Detection of Android and IoT Malware

5.1 Summary of Completed Work

Our third research presented an innovative approach using machine learning models, specifically Random Forests and Graph Neural Networks. This was done to classify malware and benign samples effectively in Android and ELF environments. Utilizing two distinct datasets, we demonstrated that GNN models outperform traditional RF models in detecting complex malware patterns. The study's findings enhance our understanding of malware dynamics but also offer robust solutions for improving detection accuracy in increasingly complex digital environments. Our work has significant implications for the development of security measures for both AR/VR and the broader Android platforms.

5.2 Introduction

As technological advancements accelerate and global interconnectivity increases, malware in mobile devices and server-side applications has surged, posing significant challenges to cybersecurity. Android, the most widely used mobile operating system, and ELF (Executable and Linkable Format) binaries, common in Linux-based servers, are key targets for cyber attackers. Both platforms' reliance on system-level API (Application Programming Interface) calls makes these interactions critical for identifying benign or malicious applications. Recent studies emphasize the importance of analyzing these API calls in combating cyber threats [88].

Android offer valuable insights into how applications behave, through the analysis of API calls. Malicious applications, for instance, frequently rely on system libraries, request excessive permissions, and use encrypted strings to conceal their operations, presenting a clear contrast to the patterns typically observed in benign applications [88]. Similarly, ELF binaries exhibit distinctive API call signatures that can be harnessed for malware detection [98].

However, the task of malware detection (and classification, in general) becomes more difficult with the increasing sophistication of malware, which utilizes advanced techniques such as obfuscation, encryption, and other methods to evade detection mechanisms [89]. Traditional malware detection methods have become inadequate at addressing current malware strategies, necessitating more adaptive solutions. This research leverages deep learning models to analyze API interactions comprehensively, aiming to develop scalable and precise malware detection systems. This approach spans Android and ELF platforms and provides a unified framework capable of addressing the diverse malware landscape [91]. To address the shortcomings of the existing work that does not consider evasion techniques in the machine learning-based techniques that utilize API calls, we

have several additional features, including encrypted strings, reflection usage, methods overload, static initializers, method count, large method count, and suspicious libraries usage. These features are used with traditional API calls.

This paper proposes an advanced malware detection framework that utilizes API calls from both Android and ELF binaries to train state-of-the-art machine learning models, such as GNNs and transformer-based architectures. For Android, the incorporation of additional features like encrypted strings, suspicious libraries, and hash values significantly augments detection capabilities. In contrast, for ELF binaries, the focus remains predominantly on API behavior [88,98]. These enhanced models capture malware's nuanced behaviors, increasing resilience to evasion techniques.

5.3 Methodology

This section describes the methodology for processing and analyzing two datasets: Android apps and ELF files. Both datasets are divided into benign and malicious samples, with API calls and additional features extracted for malware detection. The steps involved in extracting and categorizing the API calls and the additional features for Android are explained in detail as shown in Figure 5.

The malware detection pipeline depicted in Figure 5 encapsulates a comprehensive approach to identifying malicious software within Android applications and ELF binaries. The process is structured into five distinct stages, each contributing critically to the development and refinement of machine learning models specifically tailored for security applications. The initial stage involves the systematic collection of both benign and malicious software. For Android applications, this encompasses gathering a dataset from diverse sources, ensuring a representative sample of the current Android ecosystem. Similarly, ELF files, both benign and malicious, are compiled to form a dataset reflective of potential security threats on various systems. This stage is crucial for establishing a foundation upon which subsequent analytical processes are built.

With the datasets in place, the next phase is extracting key features necessary for effective model training. For Android apps, APKtool is employed to decompile the files. This is followed by using the `grep` command to search through the Smali code and extract pertinent information. This process allows for a granular examination of the underlying code structure, which is essential for identifying features indicative of malicious or benign intent. In contrast, Radare2 is used to extract API calls. These calls are indicative of behavioral patterns linked to malicious intent.

Following extraction, the data undergoes rigorous cleaning and preparation processes. This stage refines the data into a format optimally suited to processing by machine learning algorithms. It involves removing redundancies, handling missing values, and encoding categorical data. This ensures the quality and usability of the data for the next stages. The prepared data is then fed into two types of machine learning models: RF and GNN. Each model is trained and tested using a split of the processed data. This allows for the evaluation of each model's efficacy in detecting malware within the dataset. This stage is critical as it directly influences the effectiveness of the malware

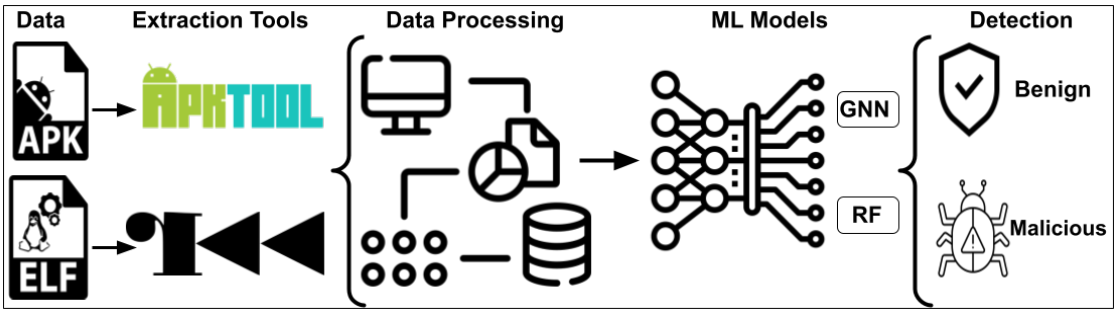

Figure 5: A visual representation of the malware detection pipeline, illustrating the integration of API extraction tools, data processing stages, and machine learning model implementations for detecting malicious software.

detection system. Performance metrics derived from this phase are used to gauge the training process's success. In the final stage, results from both the RF and GNN models are collected and analyzed to assess their predictive accuracy. The outcomes determine the models' capabilities to classify Android applications and ELF files as benign or malicious. This stage is vital for providing insights into the models' operational effectiveness and guiding future improvements in malware detection methodologies.

5.3.1 Dataset Overview

We worked with two primary datasets: **Android Apps.** This dataset consists of 799 benign and 371 malicious Android apps, including 406 AR/VR apps from [14], where 390 are benign apps and 16 are malicious apps. **ELF Files.** This dataset includes 39,788 benign ELF files and 814 malicious ELF files from [65]. For both datasets, API calls were extracted using APKtool and Radare2 and categorized. The details of these processes are provided below.

5.3.2 API Extraction and Grouping

For both the Android and ELF datasets, API calls were extracted and categorized into ten functional groups. In the case of Android apps, the APK files were decompiled using APKtool, a tool that converts APKs back into readable source code. After decompiling, `grep`, a command-line utility for searching plain-text data, was used to extract the class definitions. These extracted classes were filtered against the official Android Class API [39], which is a collection of predefined classes and functions that developers use to interact with the Android operating system. Only classes listed on the official site were retained for the study. For ELF files, Radare2 was utilized to extract the API calls, which were then cleaned to remove irrelevant or redundant information. In both datasets, the API calls were categorized into ten groups based on their functionality:

✦**Networking.**: It is API calls related to advanced secure network communication, including establishing network sockets (endpoints for sending and receiving data) and handling various critical network protocols such as HTTP, TCP/IP, and UDP.

✦**File I/O.**: File I/O is functions that handle input and output operations for files. These operations include opening, reading from, writing to, and closing files. In addition, they perform essential file management tasks like copying or deleting files.

✦**Memory Management.**: It includes API calls that are responsible for effectively managing a program's memory usage. Allocating and precisely managing memory during runtime are among the most common functions called during this process, as well as optimizing memory usage to prevent memory leaks.

✦**Process Control.**: It is related to functions that manage the execution of programs, including creating new processes, controlling their execution, and terminating processes.

✦**String Manipulation.**: This group includes API calls involved in managing text data, including functions that concatenate, format, parse, or transform strings (sequences of characters).

✦**System Interaction.**: It includes functions and operations that allow the application to interact with the underlying operating system, such as accessing system resources, performing system calls, or interacting with system hardware.

✦**Standard Library.**: It is a collection of common functions and operations provided by the programming language or operating system, such as mathematical functions, input/output operations, and utility functions for general-purpose programming.

✦**Synchronization and Thread Management.**: This group is related to API calls that manage multi-threaded applications, including creating and managing threads (lightweight processes), and ensuring synchronization to prevent data corruption when multiple threads access shared resources simultaneously.

✦**Signal Handling.**: It is functions and operations that handle system signals, which are asynchronous notifications sent to a process to indicate events such as interrupts, exceptions, or other conditions that need immediate attention.

✦**Miscellaneous.**: It includes API calls that do not fit neatly into the above categories but are still crucial for the application's functionality, often serving specialized or uncommon purposes.

The number of API calls for each application (or ELF file) was averaged per sample to facilitate the analysis. This approach provided a standardized method for comparing the usage of different types of API calls across the datasets.

5.3.3 Additional Features in Android Apps

In addition to extracting and categorizing API calls, several additional features were specifically extracted from Android apps to improve the detection of malware. These features help identify patterns and techniques commonly used by malicious applications to evade detection or execute harmful actions:

✦ **Encrypted Strings.** The presence of strings that are encrypted or obfuscated. Malware often encrypts strings to hide URLs, file paths, or other sensitive and crucial data, making it more difficult for security tools to detect malicious behavior.

✦ **Reflection Usage.** Reflection refers to the ability of a program to inspect and modify its own structure at runtime, including dynamically invoking classes, methods, or fields. In malware, reflection is often used to hide or delay the execution of malicious code by bypassing traditional static analysis methods.

✦ **Method Overloading.** This involves having multiple methods with the same name but different parameters. Method overloading can be used in malicious code to confuse detection tools, making it harder to identify specific functions and their intended purposes.

✦ **Static Initializers.** Static initialization blocks are sections of code that are executed when a class is first loaded. Malware can use these blocks to run harmful code during the app's startup, ensuring the execution of malicious activities as soon as the app is launched.

✦ **Large Method Count.** Many methods within an app may indicate the use of obfuscation techniques, such as automatically-generated code or the inclusion of suspicious libraries. This could be an attempt to overwhelm or confuse static analysis tools.

✦ **Method Count.** The count of methods in an app provides insights into its complexity. Malware may have an unusually large number of methods to perform various malicious tasks or obscure its true functionality through complex, obfuscated code.

✦ **Suspicious Libraries.** The use of third-party libraries known to be associated with malicious activities or that are commonly used for obfuscation. These libraries may include code for hiding malicious behavior, circumventing security checks, or gaining unauthorized access to sensitive data.

These additional features serve as key indicators that, when combined with the analysis of API calls, provide a more comprehensive approach to detecting malware in Android applications.

5.3.4 Separating Malware and Benign Samples

To identify the key distinguishing factors between benign and malicious samples, we analyzed the features from the ten functional groups of API calls, as well as the additional features in the Android dataset. From each dataset (Android and ELF), 200 samples were selected, comprising

100 benign and 100 malicious. The features or groups that could most effectively differentiate malicious from benign samples were used as inputs to the models.

The distribution of API calls within the ten functional groups was examined for both datasets. Statistical methods, including t-tests, were used to assess whether differences in the frequency of API calls between benign and malicious samples were significant to determine inclusion. The groups with the most pronounced differences were identified as key distinguishing factors.

In the Android dataset, additional features (section 5.3.3), such as encrypted strings, reflection usage, method overloading, *etc*, were analyzed. These features were compared in the benign samples to those in the malicious apps, and the ones that showed significant differences were selected as important indicators of malware [90]. In using this test, we aim to systematically and rigorously evaluate the most appropriate features for scalable malware detection.

By combining these distinguishing factors from both the API call groups and the additional Android features, we constructed a feature set for input into the classification models. Not all groups or features were used in the feature set, but only those that were demonstrated to be effective at discriminating between groups.

5.3.5 Learning Algorithms

Two machine learning models were employed to classify the samples as benign or malicious: RF and GNN. Both models were chosen for their distinct advantages in handling complex feature sets and relationships in malware detection.

Random Forest. RF was selected for its robustness and ability to handle high-dimensional data while mitigating overfitting through its ensemble of decision trees. Each tree in the forest is trained on a subset of data and features, and the final classification is determined through majority voting across the trees [99]. The selected API call groups and additional Android features (e.g., encrypted strings and reflection usage) were used as input for the RF model. To ensure dimensionality reduction, only features with significant discriminatory power between benign and malicious samples were included. The model was trained on 200 samples per dataset (100 benign and 100 malicious) and evaluated using standard metrics, such as accuracy, precision, recall, and F1 score.

Graph Neural Networks. GNN were chosen for their ability to model relationships between API call groups and additional Android features through graph structures. GNNs excel in capturing dependencies between features, which is critical for identifying subtle interactions in malware detection [98].

In this model, nodes represent API call groups and additional features, while edges represent their interactions. For example, connecting networking API calls and encrypted string usage could provide insight into malicious behavior. The GNN was trained using Graph Convolutional Networks, where information is propagated between nodes during training to capture local and global

data patterns. The same 200 samples per dataset were used, with evaluation methods similar to those of the RF model.

Model Comparison. RF provides a straightforward, interpretable approach due to its feature importance ranking, while GNN captures more complex relationships between features. Although both models achieved high performance, GNN was particularly effective in the Android dataset, where interactions between additional features played a significant role in malware detection.

5.4 Summary and Work to be Completed

Advanced machine learning models, specifically RF and GNN, were developed to detect and classify malware in Android and ELF environments. The focus has been on utilizing API call sequences and other relevant data features to effectively train these models. This is to enhance malware detection systems' robustness.

Work to be Completed. The next critical steps involve conducting extensive testing phases for both RF and GNN models using the collected datasets. I will evaluate the models' performance in realistic scenarios to identify necessary refinements that could optimize detection accuracy.

Following the testing, I will analyze the results, focusing on key metrics such as precision, recall, F1 score, and overall accuracy. This comprehensive evaluation will assess the models' effectiveness in accurately classifying malware and minimizing false positives. This will ensure robustness across various operational conditions.

I expect this analysis to provide deep insights into how the models function under different scenarios. It will form the basis for a detailed discussion of the outcomes. The culmination of this study will involve drafting a comprehensive conclusion that integrates the results and explores their implications. The conclusion highlights the contribution of the research and sets the stage for future advances in malware detection.

References

[1] Apps & games content ratings on google play. https://tinyurl.com/s4buprp8, — 2024. —.

[2] D. K. A., P. Vinod, S. Y. Yerima, A. Bashar, D. A. Wagner, A. T., A. Antony, A. K. Shavanas, and T. G. Kumar. Obfuscated malware detection in iot android applications using markov images and CNN. *IEEE Syst. J.*, 17(2):2756–2766, 2023.

[3] A. Abusnaina, M. Abuhamad, H. Alasmary, A. Anwar, R. Jang, S. Salem, D. Nyang, and D. Mohaisen. DL-FHMC: deep learning-based fine-grained hierarchical learning approach for robust malware classification. *IEEE Trans. Dependable Secur. Comput.*, 19(5):3432–3447, 2022.

[4] A. Abusnaina, A. Anwar, S. Alshamrani, A. Alabduljabbar, R. Jang, D. Nyang, and D. Mohaisen. Systematically evaluating the robustness of ml-based iot malware detection systems. In *International Symposium on Research in Attacks, Intrusions and Defenses, RAID*, pages 308–320. ACM, 2022.

[5] A. Abusnaina, A. Khormali, H. Alasmary, J. Park, A. Anwar, and A. Mohaisen. Adversarial learning attacks on graph-based iot malware detection systems. In *International Conference on Distributed Computing Systems, ICDCS*, pages 1296–1305. IEEE, 2019.

[6] D. Adams, A. Bah, C. Barwulor, N. Musaby, K. Pitkin, and E. M. Redmiles. Ethics emerging: the story of privacy and security perceptions in virtual reality. In *SOUPS*, pages 427–442. USENIX Association, 2018.

[7] W. U. Ahmad, J. Chi, T. Le, T. Norton, Y. Tian, and K. Chang. Intent classification and slot filling for privacy policies. *CoRR*, abs/2101.00123, 2021.

[8] W. U. Ahmad, J. Chi, Y. Tian, and K. Chang. Policyqa: A reading comprehension dataset for privacy policies. *CoRR*, abs/2010.02557, 2020.

[9] A. Alabduljabbar, A. Abusnaina, Ü. Meteriz-Yildiran, and D. Mohaisen. Automated privacy policy annotation with information highlighting made practical using deep representations. In *CCS*, pages 2378–2380. ACM, 2021.

[10] A. Alabduljabbar and D. Mohaisen. Measuring the privacy dimension of free content websites through automated privacy policy analysis and annotation. In *WWW*, pages 860–867. ACM, 2022.

[11] H. Alasmary, A. Abusnaina, R. Jang, M. Abuhamad, A. Anwar, D. Nyang, and D. Mohaisen. Soteria: Detecting adversarial examples in control flow graph-based malware classifiers.

In *International Conference on Distributed Computing Systems, ICDCS*, pages 888–898. IEEE, 2020.

[12] H. Alasmary, A. Anwar, J. Park, J. Choi, D. Nyang, and A. Mohaisen. Graph-based comparison of iot and android malware. In *Computational Data and Social Networks, CSoNet*, pages 259–272. Springer, 2018.

[13] H. Alasmary, A. Khormali, A. Anwar, J. Park, J. Choi, A. Abusnaina, A. Awad, D. Nyang, and A. Mohaisen. Analyzing and detecting emerging internet of things malware: A graph-based approach. *IEEE Internet Things J.*, 6(5):8977–8988, 2019.

[14] A. Alghamdi, A. Alkinoon, A. Alghuried, and D. Mohaisen. xr-droid: A benchmark dataset for AR/VR and security applications. *IEEE Transactions on Dependable and Secure Computing*, pages 1–13, 2024.

[15] B. Andow, S. Y. Mahmud, W. Wang, J. Whitaker, W. Enck, B. Reaves, K. Singh, and T. Xie. Policylint: Investigating internal privacy policy contradictions on google play. In *USENIX Security*, pages 585–602, 2019.

[16] A. Anwar, H. Alasmary, J. Park, A. Wang, S. Chen, and D. Mohaisen. Statically dissecting internet of things malware: Analysis, characterization, and detection. In *Information and Communications Security - 22nd International Conference, ICICS*, volume 12282 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2020.

[17] F. Arena, M. Collotta, G. Pau, and F. Termine. An overview of augmented reality. *Comput.*, 11(2):1–15, 2022.

[18] N. Ashtari, A. Bunt, J. McGrenere, M. Nebeling, and P. K. Chilana. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In *Conference on Human Factors in Computing Systems, CHI*, pages 1–13. ACM, 2020.

[19] S. B. Atitallah, M. Driss, and I. M. Almomani. A novel detection and multi-classification approach for iot-malware using random forest voting of fine-tuning convolutional neural networks. *Sensors*, 22(11):4302, 2022.

[20] Y. Bai, S. Chen, Z. Xing, and X. Li. Argusdroid: detecting android malware variants by mining permission-api knowledge graph. *Sci. China Inf. Sci.*, 66(9), 2023.

[21] A. Banik and J. P. Singh. Android malware detection by correlated real permission couples using FP growth algorithm and neural networks. *IEEE Access*, 11:124996–125010, 2023.

[22] S. A. S. Braghin. 4kdump: Exfiltrating files via hexdump and video capture. In *Workshop on Cryptography and Security in Computing Systems*, pages 1–6, 2019.

[23] D. Bui, K. G. Shin, J. Choi, and J. Shin. Automated extraction and presentation of data practices in privacy policies. *PETS*, 2021(2):88–110, 2021.

[24] G. Burdea and P. Coiffet. *Virtual reality technology*. John Wiley & Sons, 2003.

[25] R. Chand, S. ur Rehman Khan, S. Hussain, and W. Wang. TTAG+R: A dataset of google play store's top trending android games and user reviews. In *Software Quality, Reliability, and Security, QRS*, pages 580–586. IEEE, 2022.

[26] Y. Chang, S. F. Wong, C. F. L. Saenz, and H. Lee. The role of privacy policy on consumers' perceived privacy. *Gov. Inf. Q.*, 35(3):445–459, 2018.

[27] X. Chen, H. Yu, D. Yu, J. Chen, and X. Sun. Predicting android malware combining permissions and API call sequences. *Softw. Qual. J.*, 31(3):655–685, 2023.

[28] C. C. Cheng, C. Shi, N. Z. Gong, and Y. Guan. Logextractor: Extracting digital evidence from android log messages via string and taint analysis. *Digit. Investig.*, 37 Supplement:301193, 2021.

[29] J. Choi, A. Anwar, A. Alabduljabbar, H. Alasmary, J. Spaulding, A. Wang, S. Chen, D. Nyang, A. Awad, and D. Mohaisen. Understanding internet of things malware by analyzing endpoints in their static artifacts. *Comput. Networks*, 206:108768, 2022.

[30] J. Choi, A. Anwar, H. Alasmary, J. Spaulding, D. Nyang, and A. Mohaisen. Iot malware ecosystem in the wild: a glimpse into analysis and exposures. In *Symposium on Edge Computing, SEC*, pages 413–418. ACM, 2019.

[31] H. Darabian, A. Dehghantanha, S. Hashemi, M. Taheri, A. Azmoodeh, S. Homayoun, K. R. Choo, and R. M. Parizi. A multiview learning method for malware threat hunting: windows, iot and android as case studies. *World Wide Web*, 23(2):1241–1260, 2020.

[32] S. Dargan, S. Bansal, M. Kumar, A. Mittal, and K. Kumar. Augmented reality: A comprehensive review. volume 30, pages 1057–1080, 2023.

[33] M. de Vos and J. Pouwelse. ASTANA: practical string deobfuscation for android applications using program slicing. *CoRR*, abs/2104.02612, 2021.

[34] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186. ACL, 2019.

[35] A. Dirin and T. H. Laine. User experience in mobile augmented reality: Emotions, challenges, opportunities and best practices. *Comput.*, 7(2):1–18, 2018.

[36] L. Elluri, S. S. L. Chukkapalli, K. P. Joshi, T. Finin, and A. Joshi. A BERT based approach to measure web services policies compliance with GDPR. *IEEE Access*, 9:148004–148016, 2021.

[37] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. A. Wagner. Android permissions demystified. In *Conference on Computer and Communications Security, CCS*, pages 627–638. ACM, 2011.

[38] A. Giaretta. Security and privacy in virtual reality - A literature survey. *CoRR*, abs/2205.00208, 2022.

[39] Google. Class index, 2024.

[40] F. Guyton. *Feature Selection on Permissions, Intents and APIs for Android Malware Detection*. PhD thesis, Nova Southeastern University, Fort Lauderdale, Florida, USA, 2021.

[41] D. Harborth, M. Hatamian, W. B. Tesfay, and K. Rannenberg. A two-pillar approach to analyze the privacy policies and resource access behaviors of mobile augmented reality applications. In T. Bui, editor, *HICSS*, pages 1–10, 2019.

[42] D. Harborth and S. Pape. Investigating privacy concerns related to mobile augmented reality apps - A vignette based online experiment. *Comput. Hum. Behav.*, 122:106833, 2021.

[43] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. *CoRR*, abs/1802.02561, 2018.

[44] M. Hassen and P. K. Chan. Scalable function call graph-based malware classification. In *ACM Conference on Data and Application Security and Privacy, CODASPY*, pages 239–248. ACM, 2017.

[45] S. Hou, T. Lu, Y. Du, and J. Guo. Static detection of android malware based on improved random forest algorithm. In *2017 IEEE International Conference on Intelligence and Security Informatics, ISI*, page 200. IEEE, 2017.

[46] E. Iannone, F. Pecorelli, D. D. Nucci, F. Palomba, and A. D. Lucia. Refactoring android-specific energy smells: A plugin for android studio. In *International Conference on Program Comprehension, ICPC*, pages 451–455. ACM, 2020.

[47] J. Jang, J. Yun, A. Mohaisen, J. Woo, and H. K. Kim. Andro-profiler: Detecting and classifying android malware based on behavioral profiles. *CoRR*, abs/1606.01403, 2016.

[48] D. E. Krutz, M. Mirakhorli, S. A. Malachowsky, A. Ruiz, J. Peterson, A. Filipski, and J. Smith. A dataset of open-source android applications. In *Mining Software Repositories, MSR*, pages 522–525. IEEE, 2015.

[49] H. Lee, J. Park, and U. Lee. A systematic survey on android API usage for data-driven analytics with smartphones. *ACM Comput. Surv.*, 55(5):104:1–104:38, 2023.

[50] S. M. Lehman, A. S. Alrumayh, K. Kolhe, H. Ling, and C. C. Tan. Hidden in plain sight: Exploring privacy risks of mobile augmented reality applications. *ACM Trans. Priv. Secur.*, 25(4):26:1–26:35, 2022.

[51] D. Li, Y. Lyu, M. Wan, and W. G. J. Halfond. String analysis for java and android applications. In *Joint Meeting on Foundations of Software Engineering, ESEC/FSE*, pages 661–672. ACM, 2015.

[52] R. Li, W. Diao, Z. Li, S. Yang, S. Li, and S. Guo. Android custom permissions demystified: A comprehensive security evaluation. *IEEE Trans. Software Eng.*, 48(11):4465–4484, 2022.

[53] J. Lim, H. Yun, A. Ham, and S. Kim. Mine yourself!: A role-playing privacy tutorial in virtual reality environment. In *CHI*, pages 375:1–375:7. ACM, 2022.

[54] F. Liu, S. Wilson, P. Story, S. Zimmeck, and N. Sadeh. Towards automatic classification of privacy policy text. Technical report, 2018.

[55] P. Liu, Y. Zhao, M. Fazzini, H. Cai, J. Grundy, and L. Li. Automatically detecting incompatible android apis. *ACM Trans. Softw. Eng. Methodol.*, 33(1):15:1–15:33, 2024.

[56] S. Liu, F. Zhang, B. Zhao, R. Guo, T. Chen, and M. Zhang. Appcorp: a corpus for android privacy policy document structure analysis. *Frontiers Comput. Sci.*, 17(3):173320, 2023.

[57] X. Liu and K. Liu. A permission-carrying security policy and static enforcement for information flows in android programs. *Comput. Secur.*, 126:103090, 2023.

[58] Z. Liu, R. Wang, N. Japkowicz, H. M. Gomes, B. Peng, and W. Zhang. Segdroid: An android malware detection method based on sensitive function call graph learning. *Expert Syst. Appl.*, 235:121125, 2024.

[59] T. Mahmud, M. Che, and G. Yang. Analyzing the impact of API changes on android apps. *J. Syst. Softw.*, 200:111664, 2023.

[60] S. N. Maragkoudaki and C. Kalloniatis. Virtual reality as a mean for increasing privacy awareness: The escape room example. In *PCI*, pages 261–266. ACM, 2022.

[61] R. Mcconkey and O. Olukoya. Runtime and design time completeness checking of dangerous android app permissions against GDPR. *IEEE Access*, 12:1–22, 2024.

[62] A. Mishra, A. Kanade, and Y. N. Srikant. Asynchrony-aware static analysis of android applications. In *International Conference on Formal Methods and Models for System Design, MEMOCODE*, pages 163–172. IEEE, 2016.

[63] A. Mohammadinodooshan, U. Kargén, and N. Shahmehri. Robust detection of obfuscated strings in android apps. In *Workshop on Artificial Intelligence and Security, AISec*, pages 25–35. ACM, 2019.

[64] F. Nusrat, F. Hassan, H. Zhong, and X. Wang. How developers optimize virtual reality applications: A study of optimization commits in open source unity projects. In *International Conference on Software Engineering, ICSE*, pages 473–485. IEEE, 2021.

[65] N. Partush. Labeled-elfs, 2021.

[66] F. Pecorelli, G. Catolino, F. Ferrucci, A. D. Lucia, and F. Palomba. Testing of mobile applications in the wild: A large-scale empirical study on android apps. In *International Conference on Program Comprehension, ICPC*, pages 296–307. ACM, 2020.

[67] D. Prestat, N. Moha, and R. Villemaire. An empirical study of android behavioural code smells detection. *Empir. Softw. Eng.*, 27(7):179, 2022.

[68] A. Ravichander, A. W. Black, S. Wilson, T. B. Norton, and N. M. Sadeh. Question answering for privacy policies: Combining computational and legal perspectives. *CoRR*, abs/1911.00841, 2019.

[69] C. E. Rubio-Medrano, P. K. D. Soundrapandian, M. Hill, L. Claramunt, J. Baek, G. S, and G. Ahn. Dypoldroid: Protecting against permission-abuse attacks in android. *Inf. Syst. Frontiers*, 25(2):529–548, 2023.

[70] D. E. Rzig, N. Iqbal, I. Attisano, X. Qin, and F. Hassan. Virtual reality (VR) automated testing in the wild: A case study on unity-based VR applications. In R. Just and G. Fraser, editors, *International Symposium on Software Testing and Analysis, ISSTA*, pages 1269–1281. ACM, 2023.

[71] K. M. Sathyendra, S. Wilson, F. Schaub, S. Zimmeck, and N. M. Sadeh. Identifying the provision of choices in privacy policy text. In *EMNLP*, pages 2774–2779, 2017.

[72] A. Sengupta and S. Sivasankari. A novel approach for analysing and detection of obfuscated malware payloads in android platform using dexmonitor. In *Artificial Intelligence Techniques for Advanced Computing Applications, ICACT*, pages 1–9, 2021.

[73] F. Shen, J. D. Vecchio, A. Mohaisen, S. Y. Ko, and L. Ziarek. Android malware detection using complex-flows. *IEEE Trans. Mob. Comput.*, 18(6):1231–1245, 2019.

[74] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu. Pvdetector: a detector of privacy-policy violations for android apps. In *MOBILESoft*, pages 299–300. ACM, 2016.

[75] D. Soi, A. Sanna, D. Maiorca, and G. Giacinto. Enhancing android malware detection explainability through function call graph apis. *J. Inf. Secur. Appl.*, 80:103691, 2024.

[76] M. Speicher, B. D. Hall, and M. Nebeling. What is mixed reality? In *Conference on Human Factors in Computing Systems, CHI*, page 537. ACM, 2019.

[77] B. Sun, T. Takahashi, T. Ban, and D. Inoue. Detecting android malware and classifying its families in large-scale datasets. *ACM Trans. Manag. Inf. Syst.*, 13(2):12:1–12:21, 2022.

[78] G. G. Sundarkumar, V. Ravi, I. Nwogu, and V. Govindaraju. Malware detection via API calls, topic models and machine learning. In *IEEE International Conference on Automation Science and Engineering, CASE*, pages 1212–1217. IEEE, 2015.

[79] D. R. Thomas, A. R. Beresford, and A. C. Rice. Security metrics for the android ecosystem. In *Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM*, pages 87–98. ACM, 2015.

[80] T. Tomiyama, Q. Wu, and A. Kanai. Dynamic analysis for malware detection using API calls and memory usage with machine learning approach. In *12th IEEE Global Conference on Consumer Electronics, GCCE*, pages 475–477. IEEE, 2023.

[81] Unity Technologies. Unity engine. https://unity.com/. Accessed: July 2024.

[82] J. D. Vecchio, F. Shen, K. M. Yee, B. Wang, S. Y. Ko, and L. Ziarek. String analysis of android applications (N). In *Automated Software Engineering, ASE*, pages 680–685. IEEE, 2015.

[83] Y. Wang, Y. Wang, S. Wang, Y. Liu, C. Xu, S. Cheung, H. Yu, and Z. Zhu. Runtime permission issues in android apps: Taxonomy, practices, and ways forward. *IEEE Trans. Software Eng.*, 49(1):185–210, 2023.

[84] L. Wei, Y. Liu, and S. Cheung. Taming android fragmentation: characterizing and detecting compatibility issues for android apps. In *Automated Software Engineering, ASE*, pages 226–237. ACM, 2016.

[85] S. Wilson, F. Schaub, A. A. Dara, F. Liu, S. Cherivirala, P. G. Leon, M. S. Andersen, S. Zimmeck, K. M. Sathyendra, N. C. Russell, T. B. Norton, E. H. Hovy, J. R. Reidenberg, and N. M. Sadeh. The creation and analysis of a website privacy policy corpus. In *ACL*, 2016.

[86] R. Xu, J. Zhang, and L. Zhou. Malware API sequence detection model based on pre-trained BERT in professional domain. In *9th International Conference on Dependable Systems and Their Applications, DSA*, pages 1059–1060. IEEE, 2022.

[87] F. Yamaguchi, N. Golde, D. Arp, and K. Rieck. Modeling and discovering vulnerabilities with code property graphs. In *IEEE Symposium on Security and Privacy, SP*, pages 590–604. IEEE, 2014.

[88] H. Yang, Y. Wang, L. Zhang, X. Cheng, and Z. Hu. A novel android malware detection method with API semantics extraction. *Comput. Secur.*, 137:103651, 2024.

[89] H. Yang, Y. Wang, L. Zhang, Z. Hu, X. Cheng, and L. Jiang. EAMDM: an evolved android malware detection method using API clustering. In *22nd IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 889–895. IEEE, 2023.

[90] H. Yang, Y. Wang, L. Zhang, Z. Hu, L. Jiang, and X. Cheng. An android malware detection method using better API contextual information. In *Information Security and Cryptology - 19th International Conference, Inscrypt 2023*, volume 14527 of *Lecture Notes in Computer Science*, pages 24–36. Springer, 2023.

[91] J. Yang, H. Li, L. He, T. Xiang, and Y. Jin. Mdadroid: A novel malware detection method by constructing functionality-api mapping. *Comput. Secur.*, 146:104061, 2024.

[92] S. Yang, D. Yan, H. Wu, Y. Wang, and A. Rountev. Static control-flow analysis of user-driven callbacks in android applications. In *International Conference on Software Engineering, ICSE*, pages 89–99. IEEE, 2015.

[93] N. F. Yildiran, Ü. Meteriz-Yildiran, and D. Mohaisen. Airtype: An air-tapping keyboard for augmented reality environments. In *Conference on Virtual Reality, VR*, pages 676–677. IEEE, 2022.

[94] L. Yu, X. Luo, X. Liu, and T. Zhang. Can we trust the privacy policies of android apps? In *DSN*, pages 538–549. IEEE Computer Society, 2016.

[95] R. Yumlembam, B. Issac, L. Yang, and S. M. Jacob. Android malware classification and optimisation based on BM25 score of android API. In *IEEE INFOCOM Workshops*, pages 1–6. IEEE, 2023.

[96] P. D. Zegzhda, D. P. Zegzhda, E. Pavlenko, and A. Dremov. Detecting android application malicious behaviors based on the analysis of control flows and data flows. In *Security of Information and Networks, SIN*, pages 280–283. ACM, 2017.

[97] Y. Zhong, M. Shi, J. He, C. Fang, and Z. Chen. Security-based code smell definition, detection, and impact quantification in android. *Softw. Pract. Exp.*, 53(11):2296–2321, 2023.

[98] B. Zhou, H. Huang, J. Xia, and D. Tian. A novel malware detection method based on API embedding and API parameters. *J. Supercomput.*, 80(2):2748–2766, 2024.

[99] H. Zhu, T. Jiang, B. Ma, Z. You, W. Shi, and L. Cheng. HEMD: a highly efficient random forest-based malware detection framework for android. *Neural Comput. Appl.*, 30(11):3353–3361, 2018.

[100] S. Zimmeck and S. M. Bellovin. Privee: An architecture for automatically analyzing web privacy policies. In *USENIX Security*, pages 1–16, 2014.

[101] K. Zou, X. Luo, P. Liu, W. Wang, and H. Wang. Bytedroid: android malware detection using deep learning on bytecode sequences. In *Trusted Computing and Information Security*, pages 159–176, 2020.